MDE/T–11
User's Guide and Reference Manual

AA–M845B–TK

digital software

Microcomputer Development Environment for MICRO/T–11

MDE/T–11
User's Guide and Reference Manual

AA–M845B–TK

August 1982

This manual explains how to use the MDE/T–11 development system for the MICRO/T–11 microprocessor. Included in this manual are descriptions of MDE/T–11 hardware and software functions, explanations of how to use MDE/T–11, a description of all MDE/T–11 commands, and a list of MDE/T–11 messages.

This manual supersedes the *VAX/VMS MDE/T–11 User's Guide and Reference Manual,* AA–M845A–TE.

Operating Systems: VAX/VMS Version 3.0
RSX–11M Version 4.0
RSX–11M–PLUS Version 2.0
RT–11 Version 4.0

Software: MDE/T–11 Symbolic Debugger Version 1.0

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Northboro, Massachusetts 01532

digital equipment corporation · maynard, massachusetts

First Printing, June 1982
Revised, August 1982

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

© 1982 by Digital Equipment Corporation.
All Rights Reserved.

Printed in U.S.A.

A postage-paid READER'S COMMENTS form is included on the last page of this document. Your comments will assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

digital™

DEC	MASSBUS	UNIBUS
DECmate	PDP	VAX
DECsystem–10	P/OS	VMS
DECSYSTEM–20	Professional	VT
DECUS	Rainbow	Work Processor
DECwriter	RSTS	
DIBOL	RSX	

M19300

CONTENTS

FIGURES

TABLES

PREFACE

This manual tells you how to develop MICRO/T-11 applications using the tools provided by the MDE/T-11 development system. The manual explains the operation of the MDE/T-11 system, shows you how to make optimal use of its time-saving features, and provides a reference section in the form of detailed descriptions of all MDE/T-11 commands.

Related Manuals

You should be thoroughly familiar with the contents of the MICRO/T-11 Microprocessor User's Guide prior to reading this manual. You should also be familiar with the:

PDP-11 MACRO-11 Language Reference Manual

RSX-11M/RSX-11M PLUS Task Builder Reference Manual

RT-11 System User's Guide, Version 4.0

Overview of this Manual

This manual contains nine chapters and five appendixes.

Chapter 1 introduces the hardware and software components of the MDE/T-11 system.

Chapter 2 describes MDE/T-11 hardware functions -- emulation, memory simulation, state analysis -- and the hardware through which these functions are performed.

Chapter 3 describes the MDE/T-11 software functions that provide the user interface to MDE/T-11 hardware for in-circuit emulation.

Chapter 4 describes MDE/T-11 operator aids.

Chapter 5 explains how to use MDE/T-11 in program development. By means of a demonstration program, this chapter takes you through the steps of the program development cycle.

Chapter 6 explains how to debug with the MDE/T-11 symbolic debugger. It uses the demonstration program introduced in Chapter 5 to illustrate the use of MDE/T-11 commands in a typical debugging session.

Chapter 7 discusses the elements of the MDE/T-11 command language.

Chapter 8 describes all MDE/T-11 commands in detail and provides examples of their use.

Chapter 9 lists and explains MDE/T-11 messages that you may encounter when developing MICRO/T-11 applications. This chapter describes the format of MDE/T-11 messages and lists them alphabetically by message type.

Appendix A describes the MICRO/T-11 pause state, pause state entry, pause state execution, and pause state exit.

Appendix B describes the MICRO/T-11 programming and architectural characteristics that are relevant to MDE/T-11 operation.

Appendix C describes the procedures for using MDE/T-11 diagnostics.

Appendix D describes the MACRO-11 programming techniques you will need when using your program's symbols in debugging with MDE/T-11 and when positioning code in absolute locations.

Appendix E explains how to configure the MDE/T-11 system hardware for 120 or 240 Vac operation.

Documentation Conventions

The following conventions are used in this manual.

In examples, user inputs are in boldface red type. Boldface black type indicates commands and, in Chapter 9, system messages.

A carriage return is indicated by <RET>. Unless indicated otherwise, end all commands and command strings with a carriage return.

Terminal keys typed at the same time are indicated by slashes--for example: CTRL/C, CTRL/O, CTRL/U, and so forth.

CHAPTER 1

INTRODUCING MDE/T-11

MDE/T-11, with its host operating system--VAX/VMS, RSX-11M, or RT-11XM--is a development system for microcomputer systems based on the MICRO/T-11 microprocessor, a single-chip LSI version of the PDP-11. MDE/T-11 lets you use all the MICRO/T-11 features during real-time hardware and software debugging through in-circuit emulation, a hardware technique that provides a real-time debugging environment (see Chapter 2 for more details).

MDE/T-11 comprises symbolic debugger software and in-circuit emulation hardware. The symbolic debugger consists of:

- Control software
- Implementation software
- Communication software

The control software is down-line loaded from the host into the MDE/T-11 system and lets you control the MDE/T-11 system with commands that you input through a console terminal. The control software calls action routines to implement your commands.

The implementation software resides on the host and tells the control software what actions to take in response to your commands.

Control software communicates with implementation software over the communication line connecting the host to the MDE/T-11 system hardware. This communication is controlled by communication software residing on the host and in the MDE/T-11 system.

The in-circuit emulation hardware consists of the emulator, the memory simulator, and the state analyzer. These three modules communicate with one another over a high-speed bus interconnect in the MDE/T-11 backplane. The modules communicate with the MDE/T-11 software over an LSI-11 bus also located in the MDE/T-11 backplane.

Figure 1-1 illustrates the hardware and software components of the MDE/T-11 system.

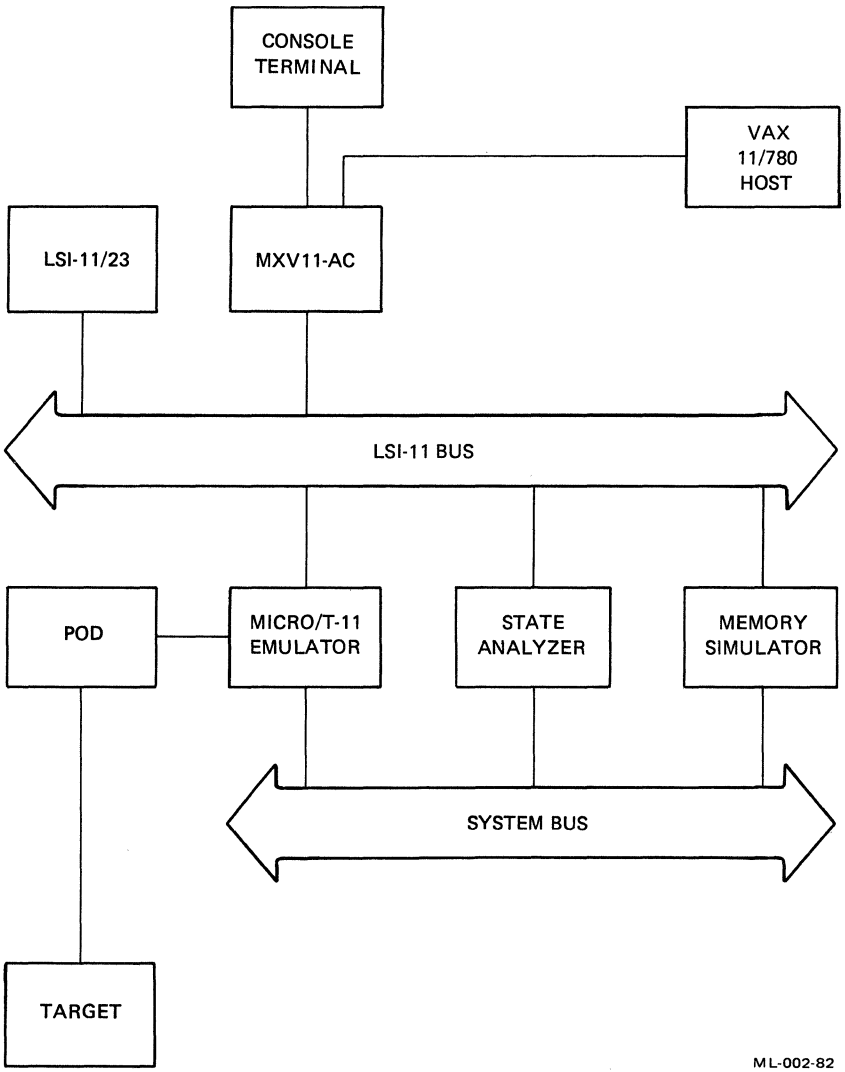


Figure 1-1 The MDE/T-11 Microcomputer Development System

In a typical development cycle without MDE/T-11 (Figure 1-2), you must build and test key circuits in your target before proceeding to the final stages of target system construction. In contrast, MDE/T-11 debugs most components of target software, including interrupt-driven software, in the absence of target hardware. (See Figure 1-3.)

Thus MDE/T-11 lets system integration take place earlier in the development cycle, eliminating many of the time-consuming steps caused by separate debugging of application hardware and software.

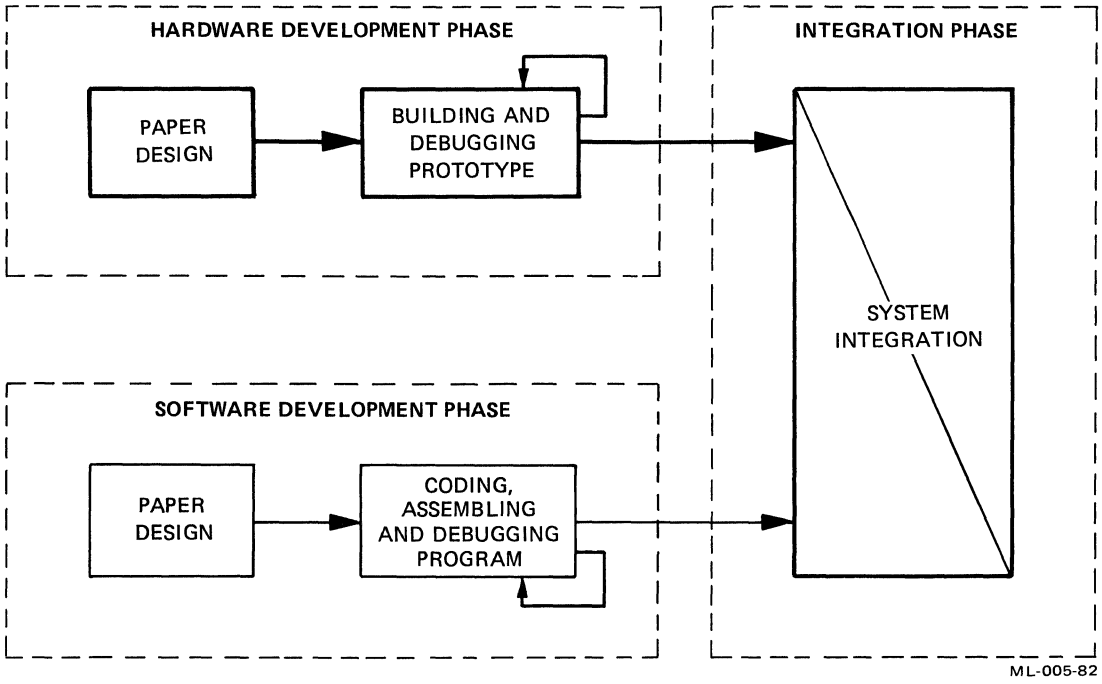


Figure 1-2 The Microcomputer Development Cycle without MDE/T-11

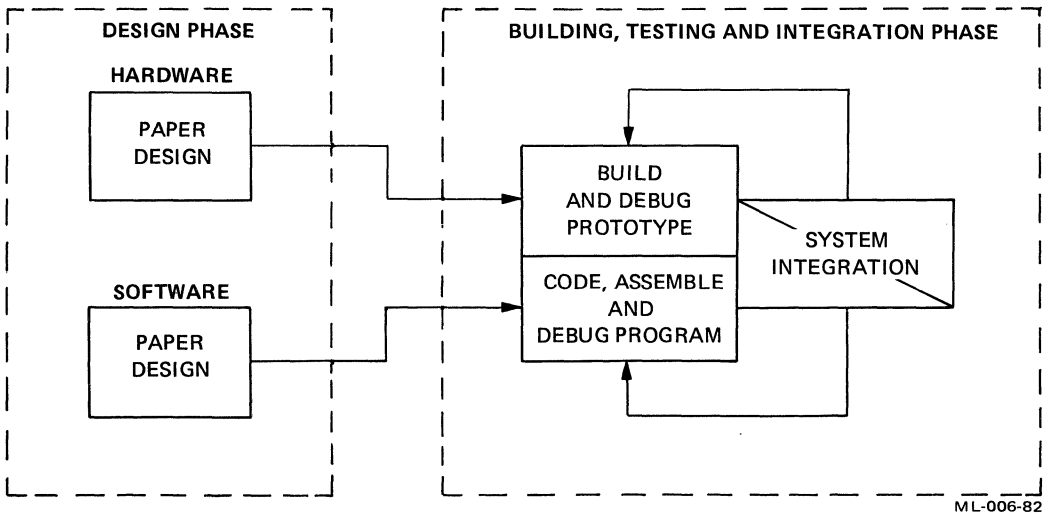


Figure 1-3 The Microcomputer Development Cycle with MDE/T-11

1.1 SYSTEM ARCHITECTURE

The hardware and software components of the MDE/T-11 system work together to provide a real-time debugging environment for use at the chip level.

1.1.1 Hardware

MDE/T-11 hardware comprises LSI-11 system components (Figure 1-1) and in-circuit emulation components (Figure 1-4).

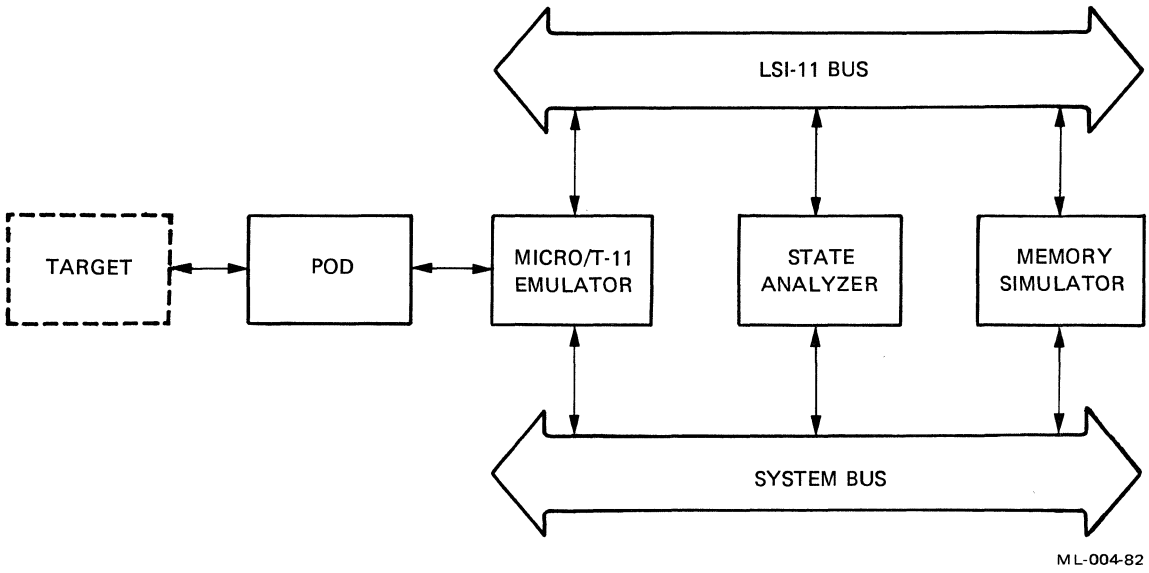


Figure 1-4 MDE/T-11 Hardware Components

System Components

- LSI-11/23 processor -- executes MDE/T-11 software and controls in-circuit emulation hardware.
- Multifunction board -- controls two serial lines, one of which interfaces to the console terminal and the other to the host system, a 32KB RAM for MDE/T-11 control software, and MDE/T-11 bootstrap ROMs.
- LSI-11 bus -- provides communication between LSI-11 system components and in-circuit emulation components.

In-circuit Emulation Components

- Emulator -- with the pod, implements timing and control for in-circuit emulation hardware.
- Pod -- contains MICRO/T-11 microprocessor and buffering logic used in in-circuit emulation.

- State analyzer -- contains four comparators, four counters, one trace RAM, and eight external logic probe lines. These components let you monitor activity on the MICRO/T-11 bus. An MDE/T-11 system can have one, two, or three state analyzers. Eight additional external probe lines can be added to the second state analyzer, for a total of 16 probe lines in an MDE/T-11 system.

- Memory simulator -- contains 32KB of memory that you can configure into MICRO/T-11 address space and provides detection of memory access violations. An MDE/T-11 system can have one or two memory simulators.

- System bus -- provides high-speed link for the emulator, state analyzer, and memory simulator and prevents bus contention between the LSI-11/23 and the MICRO/T-11 microprocessor.

1.1.2 Software

MDE/T-11 software gives you a symbolic debugger that allows access to the emulator, state analyzer, and memory simulator through a set of English commands. Application programs can be written, assembled, and linked on a host operating system (VAX/VMS, RSX-11M, or RT-11) and loaded into MICRO/T-11 address space using MDE/T-11 commands.

After loading an application program, you can debug it with the MDE/T-11's symbolic reference feature. This feature offers forward and backward translation of user-definable and global symbols. You can use a symbol instead of a value to represent an address and can display the values of addresses in "symbol+offset" format.

MDE/T-11 software also lets you implement event detection and bus cycle tracing in hardware. The event-detection mechanism lets you use both predefined events (tracepoints, watchpoints, and breakpoints) and events that you can specify (UDEs); it is much faster than software event-detection mechanisms. The bus cycle tracing mechanism lets you store MICRO/T-11 logic signals on every processor cycle of the MICRO/T-11 for display and analysis.

1.2 HOST SYSTEMS

The MDE/T-11 system is connected to a host system over a communication line through one of the serial line ports on the LSI-11/23 processor. The VAX/VMS, RSX-11M, and RT-11 operating systems can run on the host system.

The host operating system provides all file storage and development software, including editors, an assembler, and a linker or task builder utility. When you develop your application program on the host, MDE/T-11 provides transparent routing of terminal I/O between your terminal and the host system.

CHAPTER 2

HARDWARE FUNCTIONS

MDE/T-11 simulates portions of MICRO/T-11 hardware that may not be available on a target system during hardware and software development. In addition, MDE/T-11 hardware detects MICRO/T-11 bus activity and events in real time for display and analysis.

2.1 DEVELOPMENT SYSTEM

As shown in Figure 2-1, major development system components include the host system, MDE/T-11 system hardware, and the target hardware in which the MICRO/T-11 microprocessor and software are installed.

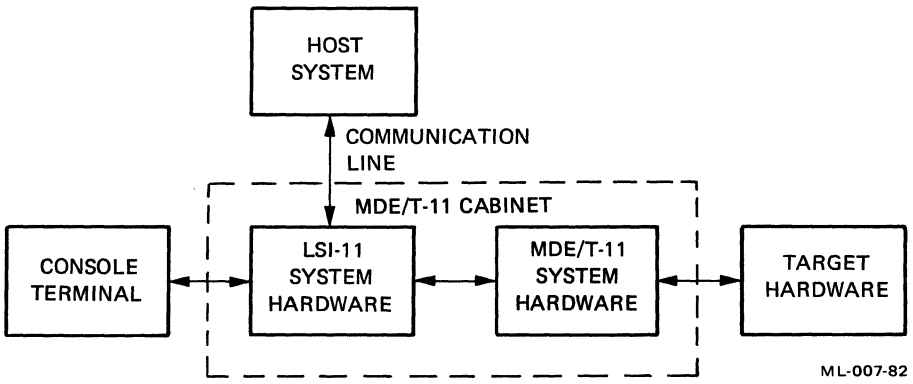


Figure 2-1 Development System Major Components

The host system provides the environment in which application programs are written, edited, assembled, and linked (also called task building). During these stages of MICRO/T-11 software development, no system hardware is involved except the console terminal and LSI-11 system hardware operating as a virtual terminal connected to the host system. The host system provides all file access for MDE/T-11 software and application programs.

A single communication line between MDE/T-11 hardware and the host system provides the interface for:

Operator commands issued to the host system

Host system messages for display on the operator's console

Down-line loading of MDE/T-11 software

Down-line loading of application software (programs and data)

Two main system components are in the MDE/T-11 cabinet: the LSI-11 system hardware and the MDE/T-11 system hardware. The LSI-11 system hardware includes a PDP-11/23 microprocessor, memory, a serial line interface, and bootstrap ROM hardware components. These LSI-11 system components perform several functions:

Provide operator access to the host system

Provide the operator MDE/T-11 command/display interface to MDE/T-11 hardware

Provide software control over the in-circuit emulation hardware

No local file storage is required. A power supply in the MDE/T-11 cabinet provides operating power for LSI-11 and MDE/T-11 system components.

The console terminal provides the operator command/display interface for the development system. MDE/T-11 software uses display features available on DIGITAL's VT100 video terminal with the VT100-AB advanced video option. It also supports DIGITAL LA120 hard-copy terminal features. Other terminals can be used, but no special display features or keypad support is provided.

The target hardware is the prototype hardware for which the application software is developed. The target is connected to the MDE/T-11 system by means of the MDE/T-11 pod. The pod includes cables that connect to the hardware in the MDE/T-11 cabinet, and a flat cable and 40-pin plug that connects to the target by means of the target's MICRO/T-11 socket.

A MICRO/T-11 microprocessor in the pod provides all normal MICRO/T-11 functions while permitting control and access by MDE/T-11 hardware functions. The cables connecting the target to the pod have minimum effect on the dynamic characteristics of the target in a real-time operating environment.

An optional user-supplied probe can be connected to the MDE/T-11 system to monitor various logic signals on the target hardware. The probe permits the monitoring of up to eight logic signals for display and analysis. A second probe can be connected to the MDE/T-11 system if it contains two or more state analyzers, for monitoring up to 16 logic signals.

2.2 MDE/T-11 SYSTEM

As shown in Figure 2-2, three MDE/T-11 system hardware components provide the main functions necessary for real-time application development: the MICRO/T-11 emulator, the memory simulator, and the state analyzer.

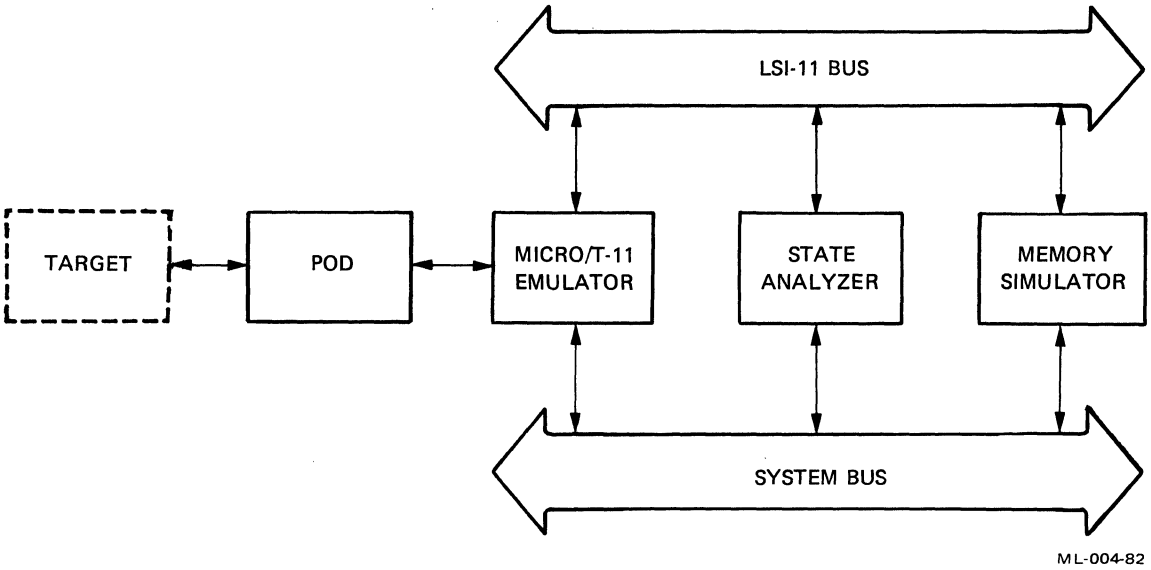


Figure 2-2 MDE/T-11 System Hardware Components

These three hardware components communicate with each other in real time over the system bus, a high-speed bus integral to the MDE/T-11 backplane. In addition, each of the three components interfaces with LSI-11 system components by means of the LSI-11 bus, providing MDE/T-11 software control and monitoring of all MDE/T-11 hardware functions.

2.2.1 **MICRO/T-11 Emulator**

MICRO/T-11 emulator hardware consists of two modules: the emulator and the pod. The pod contains a MICRO/T-11 microprocessor and has the cables and plug for connecting the target hardware to the MDE/T-11 system hardware.

The MICRO/T-11 emulator (Figure 2-3) performs in-circuit MICRO/T-11 microprocessor functions for the target hardware. These functions let you monitor and control the MICRO/T-11 and debug your target application software in real time.

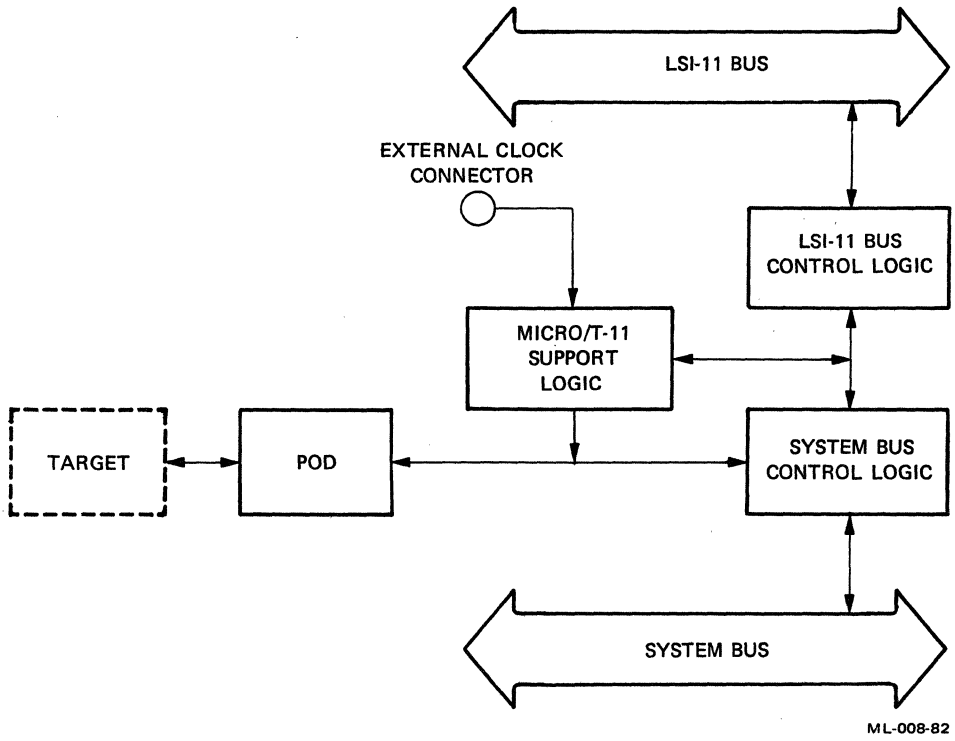


Figure 2-3 MICRO/T-11 Emulator

MICRO/T-11 support logic provides the following functions for emulating certain target hardware functions and controlling microprocessor operation.

- Processor clock source -- Circuits let you select microprocessor clock signals from one of three sources.

  Target-generated clock (selected manually via switches in the pod)

  5.Ø688 MHz MDE/T-11 system-generated clock

  External clock (connected via EXTERNAL CLOCK connector)

  Figure 2-4 illustrates the connections of the pod switches in the emulator. Refer to the description of the CONFIGURE CLOCK command in Chapter 8 for details on how to use these switches.

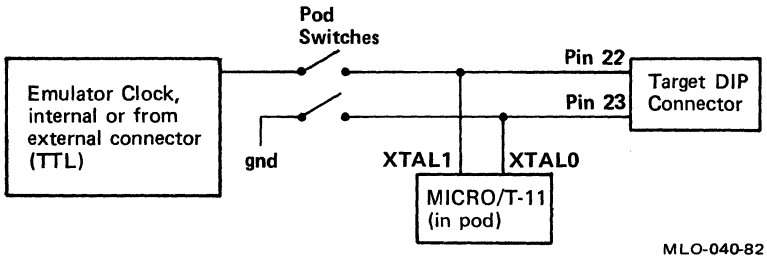


Figure 2-4 Pod Switch Connections

- MICRO/T-11 mode register source -- MICRO/T-11 mode register source bits can be selected from the target hardware or a user-programmable mode register. The user-programmable mode register lets you select:

  The MICRO/T-11 bus width (8- or 16-bit data)

  Start/Restart addresses

  Normal/Delayed and long/short bus cycles

  Static (no refresh) or dynamic (refresh) memory, including size of memory chips

  MICRO/T-11 processor clock signals or constant clock signals (half frequency)

- Timeout -- Circuits let you specify timeout detection for excessively long times between instruction fetches.

- Power-up (PUP) signal source -- You can simulate PUP signal generation by means of MDE/T-11 commands or allow assertion of PUP by the target hardware.

NOTE

The term "power up" does not refer to the presence of 5 volts on the power pin (Vcc) of the MICRO/T-11.

- Interrupt simulation -- Circuits let you select target-generated interrupts or simulated interrupts. Simulated interrupts let you specify coded priority interrupts with implicit vectors (in the MICRO/T-11 microprocessor) or external interrupt vectors. When simulated interrupts are selected, DMA cycles (produced only by target hardware) are not executed, but target-generated HALT and PF interrupts are executed in a normal fashion.

- Pause state machine -- The pause state machine (Appendix A) suspends MICRO/T-11 execution as certain events are detected or when halted by MDE/T-11 software functions. The pause state machine consists of logic circuits and routines that simulate a halt state by suspending and preserving the execution environment of the MICRO/T-11 application program.

In the pause state, no application program execution takes place, and neither DMA cycles nor interrupts are executed. For this reason, examining MICRO/T-11 bus signals with external test equipment during the pause state reveals bus activity unrelated to the application program. Consequently, analysis of this activity is neither necessary nor recommended for application program development.

NOTE

HALT and power fail (PF) MICRO/T-11 interrupts are pseudo edge sensitive; that is, an interrupt is generated whenever either signal is negated and followed by an assertion. To prevent erroneous HALT and PF interrupts resulting from pause-state machine entry and exit, both interrupt signals are latched during each MICRO/T-11 bus cycle (except when in the pause state). When in the pause state, the state of both signals is preserved. On exiting the pause state, the signals are latched again during each bus cycle, and normal HALT and PF interrupt operation is enabled.

- Single stepping -- Circuits let you single-step through program execution. At each step, execution halts (pause-state machine invoked) until you issue an MDE/T-11 command to execute the next instruction.

2.2.2 **Memory Simulator**

Memory simulator hardware consists of one or two memory simulator modules, each having 32KB of memory. Two memory simulators are required for simulating the entire MICRO/T-11 address space.

Memory simulator hardware (Figure 2-5) lets you select target and simulated memory, or either, during MICRO/T-11 application program development. You can also simulate RAM, ROM, or no memory and configure various combinations of target and simulated memory.

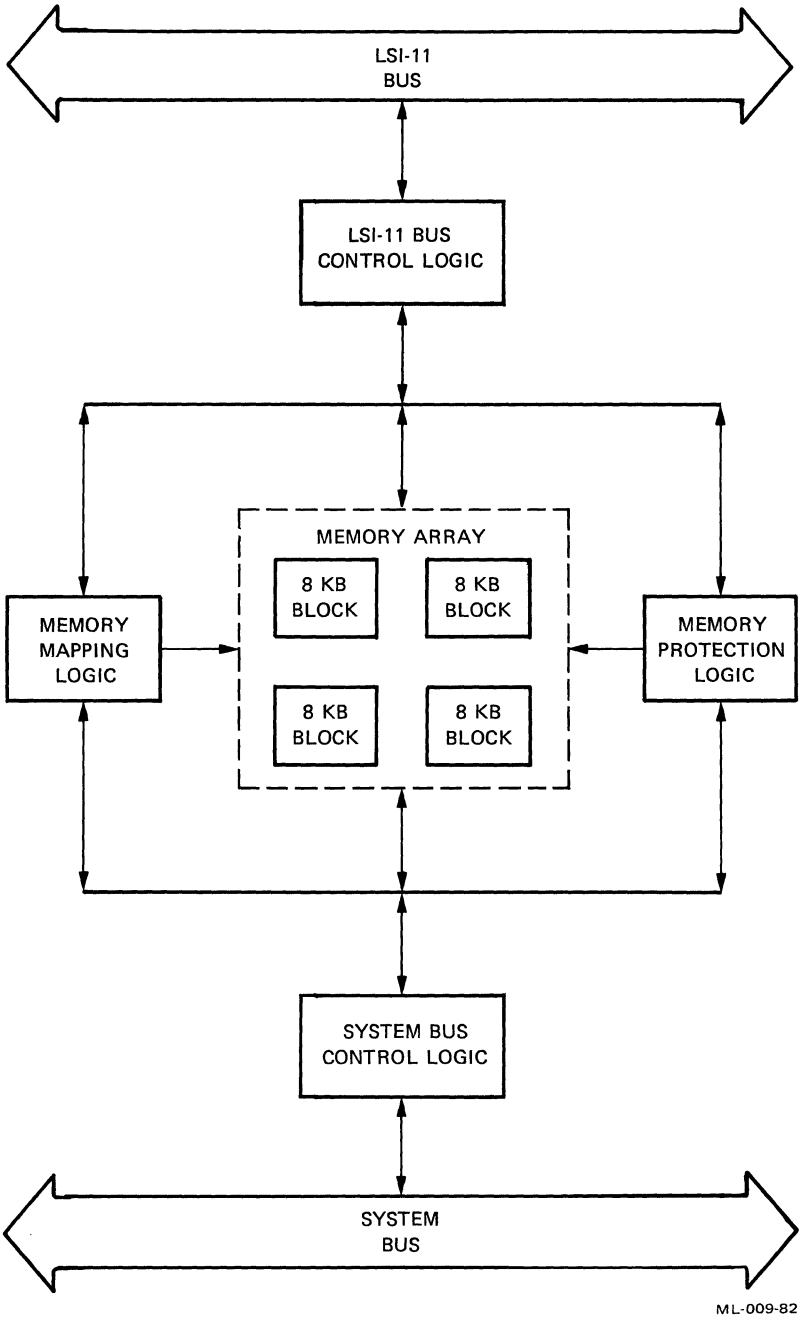


Figure 2-5 Memory Simulator Functions

The memory array consists of four 8KB blocks of RAM. Memory mapping logic responds to MDE/T-11 commands by mapping these 8KB blocks for use as simulated memory at addresses you specify. Certain guidelines must be observed regarding address boundaries, as described in Section 3.3.

Memory protection logic responds to MDE/T-11 commands by designating 256-byte increments of the 8KB block or the entire block for:

RAM (neither read nor write protected)

ROM/PROM (write protected)

Absent (both read and write protected)

Simulating target memory is useful during the early stages of hardware development. Rather than delay debugging because memory is unavailable, you can simulate target memory by allocating memory in the memory simulator and proceeding with software development.

Simulating ROM is useful for developing PROM-based application software because you can perform real-time debugging without programming and installing PROM chips on the target. Memory simulator commands let you load code and data into areas of the memory simulator or target and simulate ROM by write-protecting those areas.

During all MICRO/T-11 memory access cycles, addresses are passed to the memory simulator by means of the emulator and the system bus. The memory simulator looks at the addresses to determine if the physical memory resides in the target or the memory simulator. A signal is then passed back to the emulator so it can enable the correct data paths.

The memory simulator also looks at the protection status of the addresses and compares it with the type of bus transaction taking place. If a violation occurs, such as trying to access unmapped memory or write in write-protected memory, the error is fed back to the emulator, and emulation is halted at the next instruction fetch boundary. If the violation occurs during an instruction fetch cycle, the instruction is not executed. Memory violations detected and reported in this manner are useful when debugging ROM/PROM applications.

2.2.3 State Analyzer

Each MDE/T-11 system contains one to three state analyzers. Each state analyzer has 4 event detection circuits, permitting up to 12 events to be defined simultaneously.

Figure 2-6 illustrates the major components of a state analyzer.

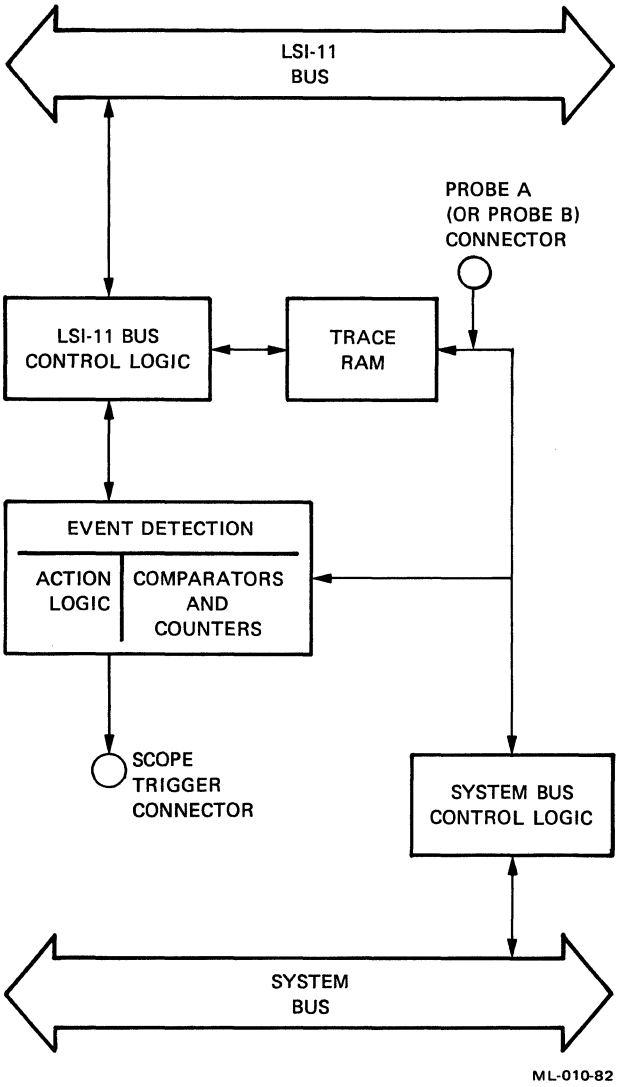


Figure 2-6 State Analyzer Functions

State analysis functions performed by the state analyzer let you monitor MICRO/T-11 application program execution, detect and report software and MICRO/T-11 hardware events that you define using MDE/T-11 commands, and capture data from the MICRO/T-11 bus for display and analysis. These functions are accomplished in hardware through event detection and bus cycle tracing. These functions are described in greater detail in the following paragraphs. External probes are discussed separately at the end of this chapter.

2.2.3.1 **Event Detection** - Event detection circuits consist of comparators and counters that monitor the MICRO/T-11 bus for predefined or user-defined events. If the conditions defining the event are met, a counter is decremented. The event is signaled only if the conditions are detected and counted a specified number of times. When the event is signaled, various actions are taken by the event detection circuits in response to parameters specified in event detection commands. These actions include placing the MICRO/T-11 in the pause state and displaying a message.

Events are defined by loading a state template (60-bit data pattern consisting of fields illustrated in Figure 2-7) into one of the four comparators on a state analyzer, and detected by comparing the state template pattern with data patterns made up of MICRO/T-11 bus signals, state analyzer flag bits, and external probe bits. When a match occurs, an event is detected.

MDE/T-11 commands let you define events and/or use three predefined events (breakpoints, tracepoints, watchpoints). In predefined events, MDE/T-11 software translates your commands into state templates that are loaded into the comparators.

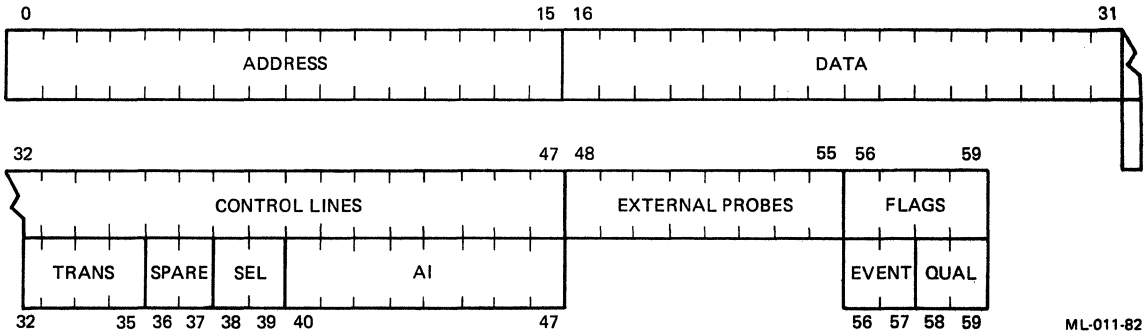


Figure 2-7 State Template Word Format

MICRO/T-11 bus signals are represented in state template bits 0 through 47. Parameters in MDE/T-11 commands let you specify the logical state or mask (don't-care) for each bit when setting user-defined events.

NOTE

TRANS bits are not taken directly from the MICRO/T-11 bus. They are encoded by the emulator to signify one of the MICRO/T-11 transaction types.

Sixteen external probe bits -- eight for state analyzer A (PROBE A) and eight for state analyzer B (PROBE B) -- specify the logical states of up to sixteen external probe lines for event detection. Each probe can monitor eight logic signals. You can use only one probe (eight probe bits) for an event. A second state analyzer is required when using a second external probe.

Four state analyzer flags are in each state analyzer. The flags are represented as four bits in the state template (bits 56 through 59). Two event flags, 0 and 1, are set automatically whenever comparator 0 and 1, respectively, declare events. Two additional flags, called qualifier flags, are set only as specified for user-defined event actions.

Flags are useful for defining events that depend on other events being signaled. Event flags can be used only for event detection within the same state analyzer. However, qualifier flags, which are set simultaneously in all state analyzers, let you set events that depend on actions in other state analyzers. Event and qualifier flags can be cleared by various means and conditions as described in Section 3.2.1.

Each comparator has a counter. The MDE/T-11 command that loads the state template into the comparator should contain a parameter that specifies a count value. Each time an event is detected, the count is decremented. If the count goes to zero, the event is signaled, appropriate MDE/T-11 action is taken, and the initial value of the count is restored.

One of the actions that can be produced when user-defined events are signaled is generating a scope trigger pulse. This pulse is available through the SCOPE TRIGGER connector on the front of the MDE/T-11 cabinet. The pulse can be used to trigger test equipment external to the MDE/T-11 system.

2.2.3.2 **Bus Cycle Tracing** - The trace RAM lets you trace each MICRO/T-11 bus cycle. The trace RAM is composed of high-speed static memory with 1Ø24 locations and 56 bits in each location. Each location stores the same information described in the state template (Figure 2-6), with the exception of flag information. Also, the information stored matches actual bus cycles as they occur in real time. You can start or stop bus cycle tracing by specifying appropriate actions for user-defined events, configure the trace analyzer to trace always, or trace only a specified number of cycles (see the CONFIGURE ANALYZER command in Chapter 8).

The trace RAM operates as a circular buffer with a pointer that indicates the first unused location as shown in Figure 2-8. When full, the trace RAM retains only the most recent 1Ø24 bus cycles; previous bus cycles are lost.

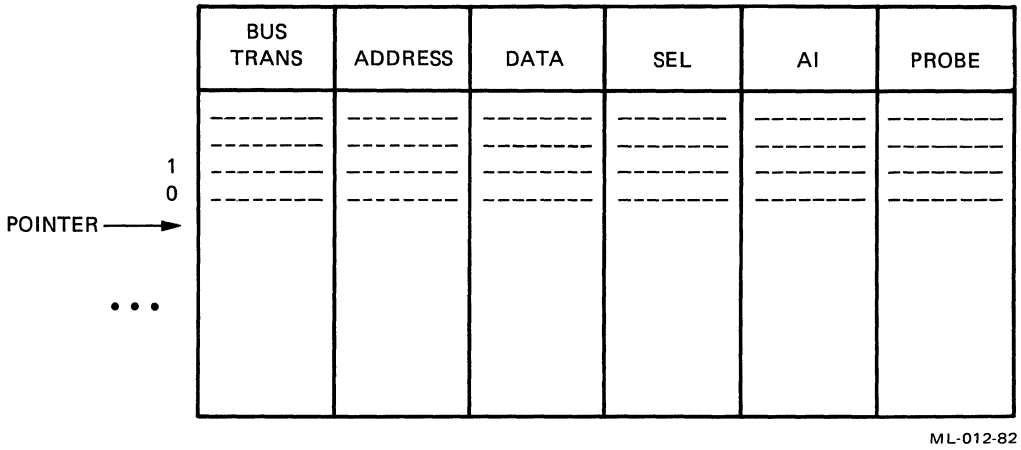


Figure 2-8 Trace RAM Fields and Addressing

2.2.3.3 **External Probes** - The PROBE A and PROBE B connectors on the front of the MDE/T-11 cabinet are for use with optional, user-supplied external probes. Each connector provides the hardware interface for one probe containing eight external lines to be monitored by one of the state analyzers. A second analyzer is required for a second probe (16 external probe lines).

NOTE

MDE/T-11 accepts Tektronix P6451 (or equivalent) data acquisition probes.

A clock attached to the probe lines latches data on the RISING or FALLING edge of the clock signal. The parameter (RISING or FALLING) that you specify in the CONFIGURE CLOCK command determines the clock edge on which probe line data is latched.

CHAPTER 3

SOFTWARE FUNCTIONS

MDE/T-11 software functions provide the user interface to MDE/T-11 hardware for in-circuit emulation, state analysis, and memory simulation. The software performs these functions in response to three groups of commands:

Emulation commands control the operation of MICRO/T-11 in-circuit emulation, including the mode of operation, starting or stopping execution, and single-stepping one or more instructions

State analysis commands detect real-time user-defined events and set breakpoints, watchpoints, and tracepoints

Memory simulation commands simulate various target system RAM and/or ROM configurations in MDE/T-11 memory simulator hardware

The following sections provide general descriptions of these commands. Refer to Chapter 8 for detailed descriptions including syntax and examples of use in a debugging session.

3.1 IN-CIRCUIT EMULATION

In-circuit emulation functions provide the user interface to MICRO/T-11 emulation hardware. You perform these functions through the following MDE/T-11 emulation commands:

CONFIGURE MODE	POWER
CONFIGURE CLOCK	HALT
CONFIGURE TIMEOUT	WAIT
SHOW CONFIGURE	SIGNAL
SHOW TARGET	

CONFIGURE MODE

The CONFIGURE MODE command lets you control the source and contents of the MICRO/T-11 emulator mode register, a 16-bit register through which all MICRO/T-11 operational features are selected. You can direct the command to configure mode bits in the mode register in the MICRO/T-11 emulator or in the mode register in the target hardware (if one is provided). The MICRO/T-11 will power up or configure itself using the mode register source specified.

CONFIGURE CLOCK

The CONFIGURE CLOCK command selects the processor clock signal from the emulator's 5.Ø688 MHz crystal-controlled source or an external source connected via the EXTERNAL CLOCK connector on the front of the MDE/T-11 cabinet. (See Figure 2-4.)

CONFIGURE TIMEOUT

The CONFIGURE TIMEOUT command turns the MICRO/T-11 emulator's processor-timeout detection feature on and off. When this feature is turned on, the emulator detects the processor's failure to assert a fetch transaction within a fixed period of time (approximately Ø.6 second).

SHOW CONFIGURE

The SHOW CONFIGURE command displays the memory, state analyzer, and MICRO/T-11 mode register configurations. The information displayed is the result of parameters set previously in CONFIGURE MEMORY, CONFIGURE ANALYZER and CONFIGURE MODE commands.

SHOW TARGET

The SHOW TARGET command displays a one-line report showing whether the MICRO/T-11 is powered up, running, or in the pause state (Appendix A). This command also displays the source of interrupts specified in the SIGNAL command.

POWER

The POWER command lets you simulate a power-on condition and direct the MICRO/T-11 to begin execution or enter the pause state. This command also lets you select the power-on initialization (PUP) signal source from the MDE/T-11 system (for simulated power on) or from the target system.

HALT

The HALT command puts the MICRO/T-11 in the pause state and displays the contents of all its registers.

WAIT

The WAIT command suspends command input from the terminal or an indirect command file until the MICRO/T-11 enters the pause state.

SIGNAL

The SIGNAL command selects the target hardware or the MDE/T-11 emulator as the source of interrupts. When simulating interrupts, you can specify encoded priority interrupts with implicit vector addresses, or you can specify external interrupt vectors. Simulated interrupts are useful when debugging interrupt service routines.

NOTE

DMA requests are ignored whenever interrupt simulation is in effect. However, power fail and halt traps issued by target hardware are executed.

3.2 STATE ANALYSIS

State analysis functions provide the user interface for event detection and bus cycle tracing. State analysis commands let you capture and analyze MICRO/T-11 bus cycle transactions in real time.

This section gives you an overview of event detection and bus cycle tracing, including a detailed description of your MDE/T-11 command interface with the state template, event and qualifier flags, and user-defined event actions.

3.2.1 Event Detection

Event detection commands let you specify particular conditions that produce a user-defined event (UDE) or three predefined events (breakpoints, tracepoints, watchpoints).

3.2.1.1 **Setting User-defined Events** - You specify a user-defined event by loading a 60-bit state template and a 60-bit mask into one of the four comparators in a state analyzer. The mask specifies which conditions or parameters the state analyzer will use when detecting events. MDE/T-11 software produces the state template and mask in response to parameters defined in the SET UDE command.

The state analyzer detects a user-defined event whenever a match occurs between the bits set in the state template and the pattern on the MICRO/T-11 bus during each processor cycle. You can direct MDE/T-11 to signal an event after each detected event or after a specified number (count) of events are detected by a state analyzer (count values range from 1 to 256). When the event is signaled, the count is automatically set to the initial value.

For each state analyzer, you can set four events, one for each comparator, and you can have up to three analyzers in the MDE/T-11 system. State analyzers are identified by the letters A, B or C. Comparators on each state analyzer are identified by the numbers 0, 1, 2 or 3. Thus you can specify the third comparator in the first state analyzer as A:2. For compatibility with other symbolic debuggers, comparators in the MDE/T-11 system are also assigned ordinals.

Comparator	Ordinal
A:0	0
A:1	1
A:2	2
A:3	3
B:0	4
B:1	5
B:2	6
.	.
.	.
.	.

3.2.1.2 **State Analyzer Flags** - Each state analyzer has four flags that signal event comparator status.

Two flags, called event flags Ø and 1, are provided on each state analyzer. They indicate the event detection state of comparators Ø and 1, respectively, in the analyzer. An event flag is automatically set whenever one of its comparators signals an event. Event flags are cleared when a RESET action occurs, when you issue a RESET ANALYZER command, or when you issue a command that sets or modifies an event for a comparator.

The remaining flags are called qualifier flags Ø and 1. They can be set by any comparator in the MDE/T-11 system, as opposed to event flags, which are set only within their respective state analyzers. Qualifier flags are set only when a user-defined event is signaled and you specified a SIGNAL QUALIFIER action in its SET UDE command. Qualifier flags are cleared by RESET actions or RESET ANALYZER commands or whenever you set or modify an event for comparators 2 or 3 in a particular state analyzer. (Qualifier flags on other state analyzers are not affected.) Consequently, you should frequently use the RESET ANALYZER command to initialize qualifier flags as required, if they are used in more than one state analyzer for event detection.

Parameters in the SET UDE command let you define the states of any event and qualifier flags for event detection conditions. The flags are useful when you want to detect a series of user-defined events.

3.2.1.3 **State Template** - The state template for each comparator (Figure 2-6) is 6Ø bits long and consists of the following fields:

16 ADDRESS bits

16 DATA bits

8 AI bits

2 SEL bits

4 TRANSaction bits that define the type of bus cycle in progress

- READ -- Read transaction
- WRITE -- Write transaction
- READ DMA -- DMA read cycle
- WRITE DMA -- DMA write cycle
- FETCH -- Instruction fetch (read transaction)
- IACK -- Interrupt acknowledge (vector read transaction)
- REFRESH -- Memory refresh cycle
- ASPI -- Assert priority in

8 EXTERNAL probe bits

2 EVENT bits

2 QUALifier bits

2 SPARE bits

3.2.1.4 **Event Actions** - All events when signaled produce one or more actions. Breakpoints, tracepoints, and watchpoints place the MICRO/T-11 in the pause state (BREAK action); however, tracepoints are followed by an implicit GO command. User-defined events can produce one or more of the following actions (specified in the SET UDE command):

BREAK	Stop target program execution (enter pause state)
TRIGGER	Generate an output pulse on the SCOPE TRIGGER connector on the front of the MDE/T-11 cabinet
TRACE	Save the data pattern for each successive bus cycle in the trace RAM starting with the next bus cycle
STOP	Stop saving the data pattern in the trace RAM following the next bus cycle
RESET	Initialize event counters and clear event and qualifier flags on all state analyzers
SIGNAL QUALIFIER	Set a qualifier flag

3.2.1.5 **Bus Cycle Tracing** - Bus cycle tracing is initiated by TRACE action parameters and stopped by STOP action parameters. When bus cycle tracing is in progress, information describing each MICRO/T-11 bus cycle is stored in sequential locations in the trace RAM. The trace RAM operates as a circular buffer with storage for 1Ø24 bus cycles. Only information for the most recent 1Ø24 bus cycles is retained in the trace RAM (except in a CONFIGURE ANALYZER command that specifies the RETAIN qualifier).

The information for each bus cycle consists of the:

Frame numbers (sequentially descending decimal numbers that identify trace RAM locations)

Bus transaction status (type)

Address of the location accessed, instruction fetched, or data written or read

State of the SEL and AI lines

External probe line logical states (if used)

The DISPLAY TRACERAM command lets you display any (or all) information in the trace RAM in any radix mode.

3.2.2 State Analysis Commands

State analysis commands perform the MDE/T-11 hardware functions of event detection and bus cycle tracing.

Event Detection Commands

- CONFIGURE ANALYZER
- RESET ANALYZER
- SET UDE, SHOW UDE, CANCEL UDE
- SET BREAK, SHOW BREAK, CANCEL BREAK
- SET TRACE, SHOW TRACE, CANCEL TRACE
- SET WATCH, SHOW WATCH, CANCEL WATCH

Bus Cycle Tracing Commands

- CONFIGURE ANALYZER
- DISPLAY TRACERAM
- CLEAR TRACERAM

CONFIGURE ANALYZER

The CONFIGURE ANALYZER command specifies the modes of operation of the trace RAM. This command also specifies the clocking mode for the external probe lines.

RESET ANALYZER

The RESET ANALYZER command initializes all comparator event counters and clears all event and qualifier flags.

SET UDE, SHOW UDE, CANCEL UDE

The SET UDE, SHOW UDE, and CANCEL UDE commands specify, display, and cancel user-defined events.

The SET UDE command loads the state template and mask into a comparator that describes the event and specifies a count and the action to be taken when the event occurs.

The SHOW UDE command displays the current status of a particular event that is specified by including the state analyzer (letter) and comparator (number) in the command. Information displayed includes the:

Event ordinal

Current event count

Action to be taken when the event is signaled

Relevant fields in the event mask

The CANCEL UDE command cancels the specified event.

SET BREAK, SHOW BREAK, CANCEL BREAK

The SET BREAK, SHOW BREAK, and CANCEL BREAK commands specify, display, and cancel breakpoint events.

The SET BREAK command loads a state template and mask describing the breakpoint into a comparator. Any breakpoint or tracepoint previously set for the specified location is canceled.

A breakpoint is a predefined event that occurs after an instruction at an address specified in a SET BREAK command is executed. A breakpoint occurs only if the detected event occurs the number of times stated in the COUNT and/or AFTER parameters. The breakpoint causes the MICRO/T-11 to enter the pause state immediately following instruction execution, and a breakpoint message is displayed.

The SET BREAK command can include the COUNT parameter to produce breakpoints only after a specified number of events are detected by the state analyzer. COUNT has the advantage of counting events in real time (the MICRO/T-11 does not enter the pause state until the count is reached), but it is limited to a maximum count of 256.

NOTE

The COUNT parameter initializes a hardware-implemented event counter in the state analyzer. Thus an event is reported to MDE/T-11 software only when the specified number of events are detected.

In addition to COUNT, the SET BREAK command can include the AFTER parameter which initializes a software-implemented counter to a specified value. AFTER causes the breakpoint to occur only when the specified number of events reported by the state analyzer occurs. The AFTER counter has the advantage of extending the event count to 65535; a second advantage is that the current software count of events can be displayed using the SHOW BREAK command. In addition, you can include both the COUNT and AFTER parameters in a SET BREAK command. When both parameters are included, they extend the event count for breakpoints to a maximum of 16,776,960. For example

```
SET BREAK/COUNT:256/AFTER:4 1000
```

As a result of this command, a breakpoint occurs at location 1000 once every 1024 times the instruction is executed.

NOTE

MDE/T-11 software execution time for decrementing the AFTER count and resuming MICRO/T-11 execution is approximately 0.5 second. Thus the AFTER parameter should be used sparingly. Do not use this parameter in critical real-time applications.

SET TRACE, SHOW TRACE, CANCEL TRACE

The SET TRACE, SHOW TRACE, and CANCEL TRACE commands specify, display, and cancel tracepoint events in the same manner as described previously for breakpoints.

Tracepoints differ from breakpoints only when the event is reported; the MICRO/T-11 does not remain in the pause state. Following display of the tracepoint message, MICRO/T-11 execution is resumed automatically.

SET WATCH, SHOW WATCH, CANCEL WATCH

The SET WATCH, SHOW WATCH, and CANCEL WATCH commands specify, display, and cancel watchpoint events in the same manner as described previously for breakpoints, except that the message content is different.

A watchpoint is a predefined event that occurs when data is written in a location specified in a SET WATCH command. A watchpoint occurs only if the detected event occurs the number of times stated in the COUNT and/or AFTER parameters.

CLEAR TRACERAM

The CLEAR TRACERAM command clears the trace RAM and initializes the RETAIN value if it is specified in a CONFIGURE ANALYZER command.

DISPLAY TRACERAM

The DISPLAY TRACERAM command displays the contents of the trace RAM during bus cycle tracing. Command parameters let you start the display with any trace RAM location and specify the:

Number of bus cycles to be displayed

MICRO/T-11 bus address display type

Radix for various display fields

NOTE

You can display any part (or all) of the trace RAM when the MICRO/T-11 is in the pause state. When the MICRO/T-11 is running, access to the trace RAM is limited, and events are disabled for a short time after displaying the contents of the trace RAM.

3.3 MEMORY SIMULATION

Memory simulation functions provide the user interface to memory simulator hardware. You perform these functions through MDE/T-11 memory simulation commands that let you:

Map and protect target memory

Load programs, data, and symbols into memory

Examine and deposit memory locations or MICRO/T-11 registers

Copy memory contents from one part of memory to another

Memory simulation lets you debug application software with or without memory on the target hardware. In addition, memory simulation permits ROM simulation, eliminating the need for programming or reprogramming ROM chips during program development stages.

You accomplish memory simulation by using the following commands:

- CONFIGURE MEMORY
- LOAD
- COPY
- DEPOSIT
- EXAMINE

CONFIGURE MEMORY

The CONFIGURE MEMORY command lets you map MICRO/T-11 address space in 256-byte increments in the target memory or the memory simulator. The MICRO/T-11 accesses memory configured in the target via the pod. Memory configured in the memory simulator is accessed directly by the MICRO/T-11. You can protect each 256-byte increment of memory from read or write access or both (for absent memory). Any attempt by your application program to access memory configured as absent or to write to write-protected memory causes the emulator to enter the pause state, and an error message informing you of the memory addressing error is displayed.

Each memory simulator contains four 8KB blocks of memory. You can map each block to any of the eight 8KB segments in the MICRO/T-11 address space. Therefore, if you have only one memory simulator in your MDE/T-11 system, you can configure simulated memory in only four 8KB segments within the MICRO/T-11 address space. If you have a second memory simulator, this restriction is removed.

NOTE

The use of any portion of a memory simulator block allocates the block for the entire 8KB address segment; the unused portion cannot be mapped to addresses outside that segment. Thus, when simulating memory with only one memory simulator, you must restrict memory simulation to addresses within four 8KB segments. This restriction is removed if a second memory simulator is used.

The following example illustrates how you can use CONFIGURE MEMORY commands to map 3 1/2 blocks of simulated memory and 4 1/2 blocks of target memory as shown in Figure 3-1.

```
MDE>CONF MEM FROM Ø to 37777 SIM<RET>

MDE>CONF MEM FROM 4ØØØØ to 47777 TARGET<RET>

MDE>CONF MEM FROM 5ØØØØ to 57777 SIM NOWRITE<RET>

MDE>CONF MEM FROM 1ØØØØØ to 137777 TARGET NOWRITE<RET>

MDE>CONF MEM FROM 16ØØØØ to 177777 SIM<RET>
```

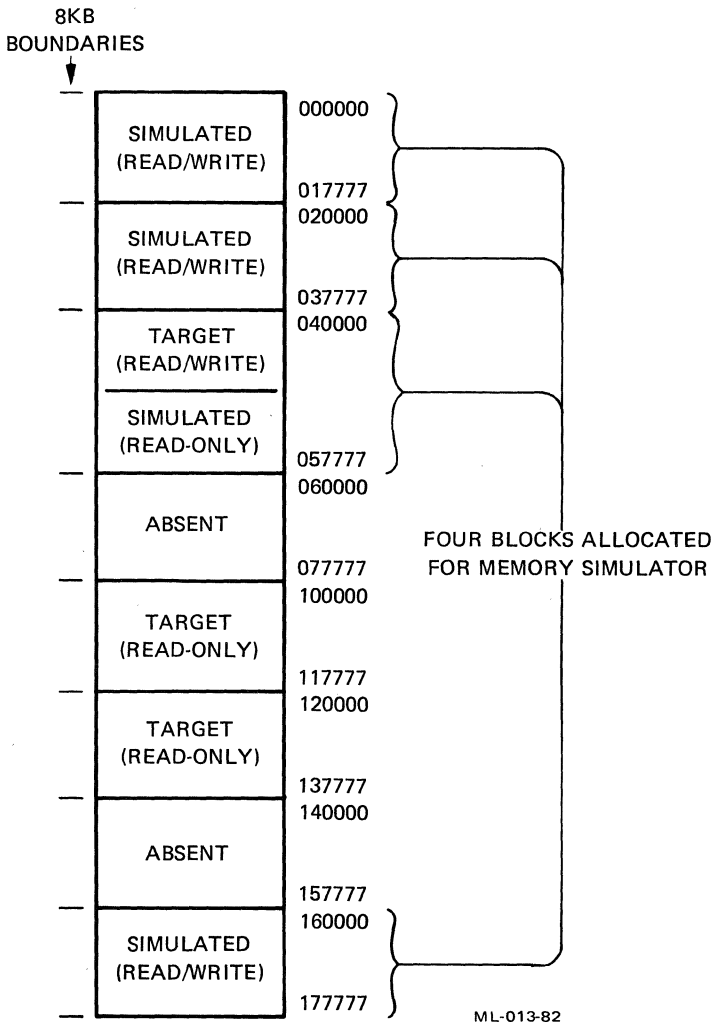


Figure 3-1 Memory Configuration Example

After issuing the commands shown above, the following command would not be allowed if your MDE/T-11 system has only one memory simulator:

MDE> **CONF MEM FROM 60000 to 60377 SIM<RET>**

The preceding command would not be allowed because a fifth 8K byte simulator block is required.

However, the command below would be allowed:

MDE> **CONF MEM FROM 40000 to 47777 SIM<RET>**

This command is allowed because it makes use of an 8K byte block of simulator memory that is already mapped.

LOAD

The LOAD command transfers your programs and data from disk files to MICRO/T-11 address space. This command also reads global program symbols into the MDE/T-11 symbol table and sets the MICRO/T-11 PC to the transfer address (if specified).

COPY

The COPY command lets you copy memory contents from one physical area to another (for example, from ROM in the target to RAM in the memory simulator) and execute an implicit CONFIGURE MEMORY command. This command also lets you copy data or code from one part of MICRO/T-11 address space to another. All copy operations are performed in 256-byte increments.

DEPOSIT

The DEPOSIT command lets you write or change the contents of a specified memory location or MICRO/T-11 register, or a range of memory locations or MICRO/T-11 registers in any radix mode and format. You can also use this command to fill a range of locations or MICRO/T-11 registers with the same value or pattern of values.

EXAMINE

The EXAMINE command lets you display the contents of a specified memory location or MICRO/T-11 register or a range of memory locations or MICRO/T-11 registers in several display formats and radices. EXAMINE is useful for disassembling program segments.

CHAPTER 4

MDE/T-11 OPERATOR AIDS

This chapter describes a group of operator aids that make the MICRO/T-11 hardware and software development tools easier to use in the debugging phase of program development. The MDE/T-11 operator aids comprise:

- Terminal support
- Indirect command files
- Log files
- Help facility
- Error reporting

4.1 TERMINAL SUPPORT

MDE/T-11 gives you general software support for non-DIGITAL terminals and some DIGITAL terminals and special software support for DIGITAL'S VT100 and LA120 terminals.

The general software support lets you use MDE/T-11 with any ASCII terminal that has an RS232 serial, asynchronous interface. However, the terminal you use must operate at or above the baud rate of the communication link between the host and MDE/T-11; maximum baud rate is 9600.

MDE/T-11 special terminal support consists of:

- Special character support
- Command keypad support
- VT100 display support

4.1.1 Special Character Support

MDE/T-11 interprets certain terminal characters as control characters on DIGITAL and non-DIGITAL terminals. Control characters correct typing errors, interrupt command execution, and stop terminal output. You execute some of these control characters by pressing the CTRL key and a letter key simultaneously--for example, CTRL/C. You execute other control characters by pressing a single key--for example, DELETE or RUBOUT.

Table 4-1 lists the control characters and their functions.

Table 4-1
Control Characters

Character	Function
CTRL/C	Interrupts execution of the EXAMINE, DISPLAY TRACERAM, and LOAD commands. Aborts and closes any active command files, aborts any active user-defined keys, and cancels command input (see CTRL/U). Echoes as ^C. Some delay may occur in aborting commands, however.
CTRL/Q	Enables terminal output that has been disabled with CTRL/S. Does not echo to the terminal.
CTRL/R	Retypes the current command line and continues to accept command input after the last character that preceded the CTRL/R. Echoes as ^R.
CTRL/S	Disables terminal output. You can reenable terminal output by typing CTRL/Q. Does not echo to the terminal.
CTRL/U	Deletes an entire line of text. Echoes as ^U.
DELETE or RUBOUT	Cancels the character just typed. Pressing it a second time cancels the next-to-last character typed, and so on.

MDE/T-11 accepts only the control characters listed above.

4.1.2 Command Keypad Support

MDE/T-11 supports a command keypad feature on the DIGITAL VT100 and LA120 terminals. This feature lets you use the keys on the keypad to define and execute MDE/T-11 commands.

4.1.2.1 **Predefined Keys** - Fourteen frequently used MDE/T-11 commands have been assigned to keys on the keypad (Figure 4-1). You can execute these commands by pressing a single key. When you enter predefined commands by using the keypad, the commands are echoed to the terminal and are recorded in a log file, if you have created one. (See Section 4.3.)

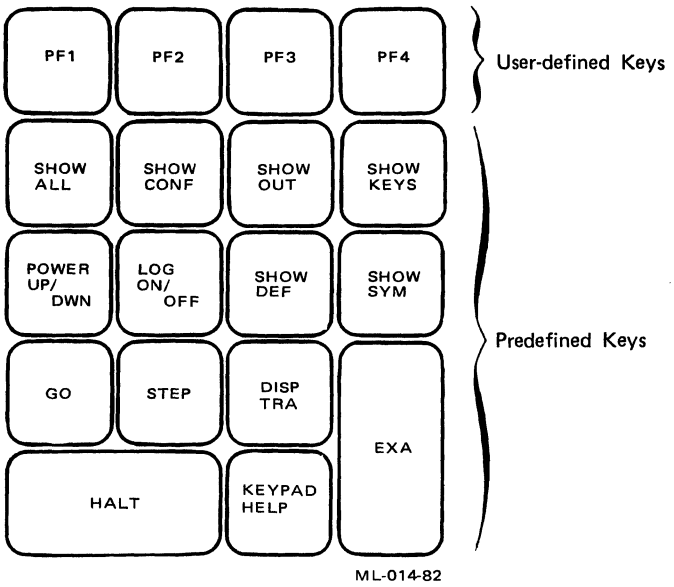


Figure 4-1 MDE/T-11 Command Keypad

All predefined keys operate identically to their command line equivalents, except for the POWER, LOG, and KEYPAD HELP keys. The POWER key turns the MICRO/T-11's power on and off; the LOG key turns logging on and off.

NOTE

When the MICRO/T-11 is powered up and running, the POWER (up/down) key does not function. This feature of the POWER key prevents you from accidentally hitting this key and powering down a running target. You must type out the POWER DOWN command to power down a running target.

The KEYPAD HELP differs from the HELP command. The KEYPAD HELP key gives you a diagram of the keypad layout, whereas the HELP command gives you a brief description of MDE/T-11 commands.

NOTE

You can type a keypad key only in response to the MDE/T-11 prompt. Keypad keys used during command line input issue illegal commands. If you type a keypad key during command line inputs, MDE rings the terminal bell and ignores the keypad key command. If you inadvertently press a keypad key while typing an MDE/T-11 command, complete or cancel the command line and then press the keypad key again.

4.1.2.2 **User-Defined Keys** - You can also assign up to eight MDE/T-11 commands to keys PF1 to PF4 with the KEYDEFINE command. You can then execute the command or commands by pressing the key you have defined. (See the KEYDEFINE command description in Chapter 8 for more information on this feature.)

4.1.3 VT100 Display Support

MDE/T-11 uses the graphics and screen-manipulation features of the VT100 terminal to provide easy-to-read displays (Figure 4-2).

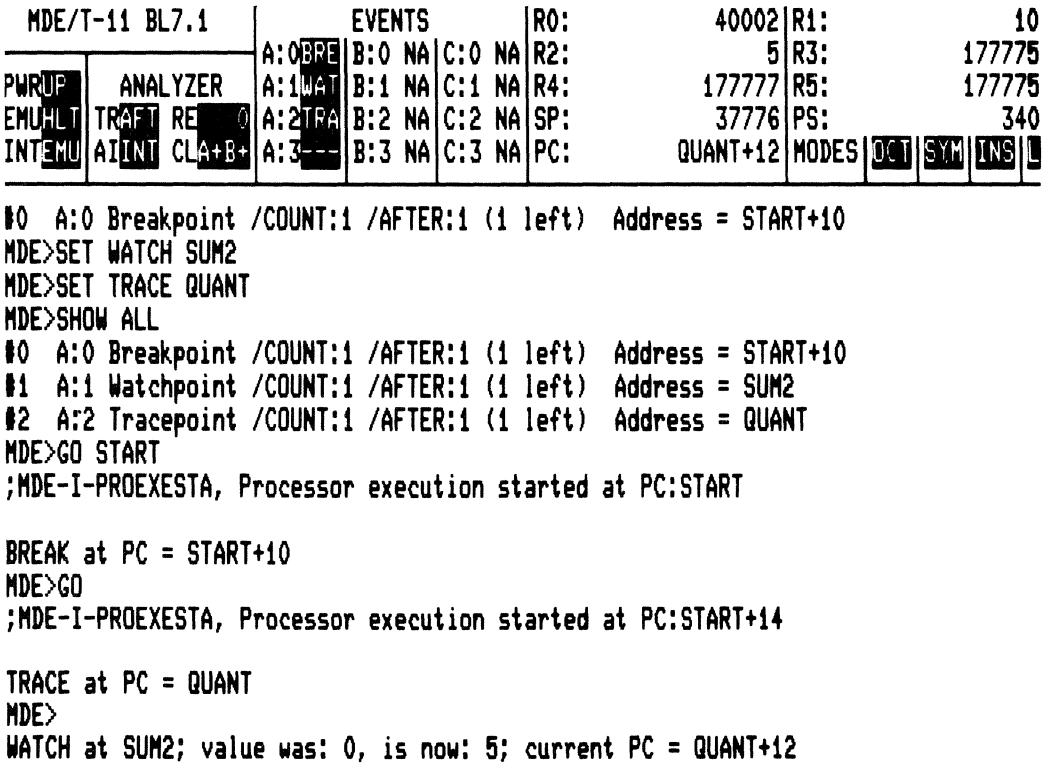


Figure 4-2 VT100 Screen Format

The MDE/T-11 control software divides the VT100 screen into two regions. The static region displays the status of the MDE/T-11 system and the MICRO/T-11 microprocessor; the scrolling region displays command input.

4.1.3.1 **Status Information** - The static region displays status information about:

- Power and run state of the MICRO/T-11 microprocessor
- State analyzer configuration
- Comparator in use

- Contents of the MICRO/T-11 registers (updated on request or when execution halts)
- Display modes
- Logging status
- Load progress

The LOAD progress display shows the relative block number of the file being loaded. During the load procedure, this display replaces the interrupt source display in the lower left corner of the static region.

The VT100-AB has a highlighting feature not available on the VT100-AA. By displaying active states and nondefault settings, such as processor running or hexadecimal display mode, this feature makes it easier to interpret the display.

4.1.3.2 **Command Input Information** - The scrolling region on the VT100 lets you see commands as you execute them, as well as the reports produced by commands. This feature, combined with the split screen, lets you see the effect of MDE/T-11 commands on the debugger and the MICRO/T-11 microprocessor immediately.

4.1.4 Activation of Special Software Support

The special software support for DIGITAL terminals is activated when you start MDE/T-11. When you issue the RUN MDE/T-11 command, MDE/T-11 sends the ANSI Device Attributes (DA) escape sequence to your terminal. If your DIGITAL LA120 or VT100 terminal responds to the DA escape sequence, MDE/T-11 turns on the special terminal-support features.

If MDE/T-11 receives no response or an unknown response from your terminal, MDE/T-11 operates in its default mode, with special terminal support turned off. You can use the SET TERMINAL command to turn the terminal support features on and off manually.

NOTE

Your VT100 does not respond to the DA escape sequence while operating in the VT52 compatibility mode. You must operate your VT100 in ANSI mode to turn on terminal support automatically. (See the VT100 User's Guide for further details.)

4.2 INDIRECT COMMAND FILES

To simplify the typing of commands, MDE/T-11 lets you place several commands in a file and execute them with the @ character followed by the file name.

There are two types of indirect command files. General indirect command files let you execute a set of frequently used MDE/T-11 commands; start-up initialization files let you automatically execute commands to start a debugging session.

4.2.1 General Indirect Command Files

MDE/T-11 lets you specify general indirect command files to execute groups of MDE/T-11 commands. For these files, you need only supply a command file name, since MDE/T-11, by default, supplies a file extension of .COM or .CMD.

You can nest indirect command files to three levels unless you turn on the logging mechanism, in which case you can nest indirect command files to only two levels; MDE/T-11 software maintains three channels for reading and writing ASCII files and allocates these channels as follows:

1. Any indirect command file = 1 channel
2. Logging in progress = 1 channel
3. HELP command issued = 1 channel

Under VAX/VMS and RSX-11M, make sure to use sequentially formatted ASCII files for indirect command files. All text editors and the MDE/T-11 logging facility normally produce sequentially formatted ASCII files. However, under some circumstances, you can obtain binary files in image mode that appear to be sequentially formatted ASCII files when displayed with the TYPE command. Use of files other than sequentially formatted ASCII files results in the following error message:

```
?MDE-E-CMDFILERR, Command file
read error, file closed
```

At the end of the command sequence in a file, MDE/T-11 closes the file, frees the channel, and returns to the MDE/T-11 command level from which the indirect command file was called. If the indirect command file was called from within another indirect command file, MDE/T-11 closes the inner command file and executes the next command in the outer command file. If the indirect command file was called from the terminal, MDE/T-11 returns to the terminal for its next command.

By default, commands read from an indirect command file are displayed at the terminal. If you want to prevent the command file input from being displayed, use the SET OUTPUT command with the NOVERIFY parameter.

Here is an example of an indirect command file called TEST.COM (or TEST.CMD).

```
SHOW OUTPUT
SET MODE OCTAL
```

You can execute TEST.COM by typing @TEST in response to the MDE/T-11 prompt (MDE>), producing the following display:

```
MDE> SHOW OUTPUT<RET>
    Output set to: TERMINAL, VERIFY, LOG (Log file is 'CDS.LOG')
```

```
MDE>SET MODE OCTAL<RET>

MDE>
```

4.2.2 Start-up Initialization File

Usually, you issue certain MDE/T-11 commands before you begin to debug your program. These commands can include one or more CONFIGURE commands, the POWER UP command, and a LOAD or SET command.

To simplify the MDE/T-11 start-up procedure, you can use a start-up initialization file (MDE.INI). You create this file just as you do a general indirect command file under your host operating system. When you create the start-up initialization file, the host operating system places it in your default directory. MDE.INI is then executed from this directory when you start the debugger. MDE/T-11 executes MDE.INI before prompting you for command input, however.

4.3 LOG FILES

MDE/T-11 lets you record all terminal input and output in a file by means of the SET OUTPUT command. This log file is helpful if you are going to repeat a debugging session. Instead of typing in the commands again, you can use the logging commands and an indirect command file to reproduce a debugging session.

If you are recording your debugging session with a log file, you can nest indirect command files to only two levels. In this situation, if you attempt to use the HELP command or to open a third indirect command file, your attempt will fail.

For more information on using log files, refer to the description of the SET LOG and SET OUTPUT commands in Chapter 8.

4.4 HELP FACILITY

The MDE/T-11 on-line help facility contains information on all MDE/T-11 commands and on some general topics as well. To learn what kind of information is available for MDE/T-11 commands, type HELP. To obtain information on specific commands, type HELP, followed by the topic and, if applicable, the subtopic.

For more detailed information on the use of the on-line help facility, see the description of the HELP command in Chapter 8.

4.5 ERROR REPORTING

MDE/T-11 reports error conditions, such as user errors and hardware problems, by means of error messages. In some cases, the error messages state the source or cause of an error. In other cases, the error messages help you to pinpoint the cause of an error and the way to correct it by suggesting possible causes and solutions.

Chapter 9 contains a complete list of MDE/T-11 error messages.

CHAPTER 5

PROGRAM DEVELOPMENT

This chapter tells you how to use MDE/T-11 to develop MACRO-11 programs under the VAX/VMS, RSX-11M, and RT-11XM operating systems. Separate sections list and explain the steps you perform while using MDE/T-11 with these operating systems. The explanations include a demonstration program that takes you through the steps in the program development cycle.

Before reading this chapter, however, you should review:

- The PDP-11 MACRO-11 User's Guide
- Appendix D in this manual, which describes special programming techniques for MDE/T-11 users

When developing an application program, you take the following steps:

1. Power up the MDE/T-11 system and enter the virtual terminal mode.
2. Log onto your host system (VAX/VMS, RSX-11M) or bootstrap your host system (RT-11XM).
3. Create a file for your source program.
4. Assemble your program with MACRO-11.
5. Link your program with TASK BUILDER (VAX/VMS, RSX-11M) or LINK (RT-11XM).
6. Debug your program (see Chapter 6).

Figure 5-1 shows the program development cycle.

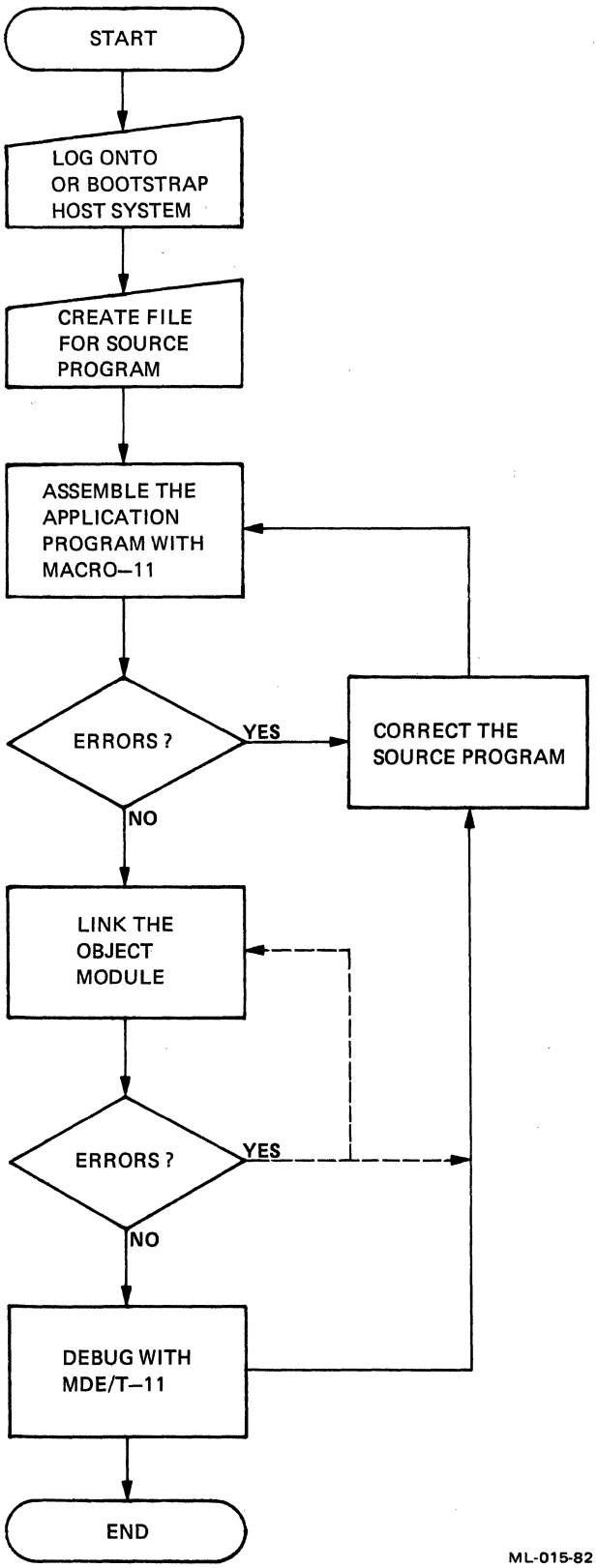


Figure 5-1 Program Development Cycle

5.1 POWER UP MDE/T-11 AND ENTER VIRTUAL TERMINAL MODE

Powering up MDE/T-11 and entering the virtual terminal mode connects the MDE/T-11 console terminal to the host under software control so you can communicate with the host. This feature is the same for all three host systems.

In the virtual terminal mode, all characters you type go to the host system, and all characters sent by the host are displayed on the MDE/T-11 console terminal. However, two characters that you type at the MDE/T-11 console cause special actions:

- BREAK -- When you press the BREAK key, this character does not assert a break on the host serial line, but causes the LSI-11/23 to halt and to enter console ODT.

- CONTROL UNDERSCORE -- When you type this character at the MDE/T-11 console terminal, the terminal sends a break to the host. On VT100 and LA120 terminals, simultaneously hold down the CTRL key and press the slash (/) key.

  On LA36 and VT52 terminals, simultaneously hold down the CTRL and SHIFT keys and press the underscore key (_).

  On any other terminal, check the applicable user's guide to determine how to send the ASCII character whose value is 37 (octal).

To power up the MDE/T-11 system and enter virtual terminal mode:

1. Put the HALT switch in the RUN (up) position (Figure 5-2).

2. Put the AUX switch in the OFF (down) position to turn off the line-time clock.

3. Turn on the POWER switch on the MDE/T-11 cabinet.

4. Toggle the RESTART switch on the cabinet.

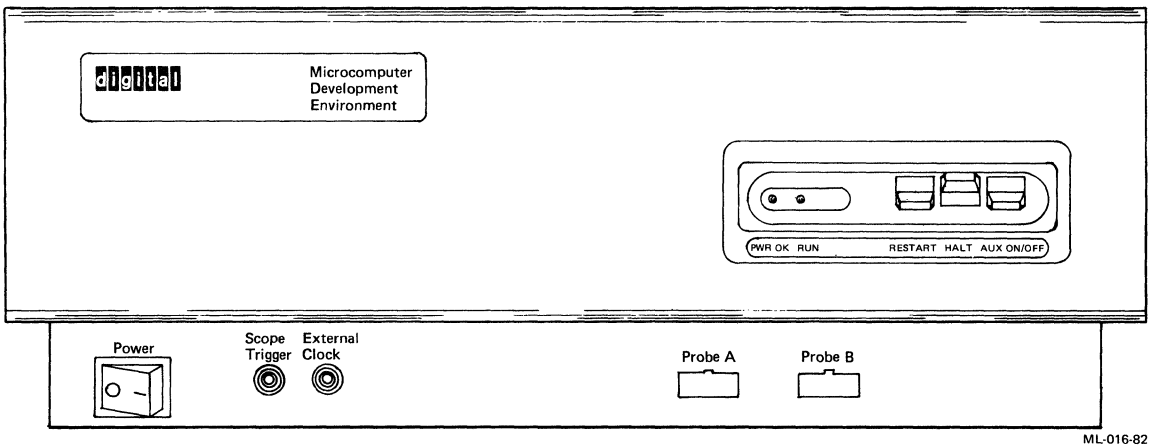


Figure 5-2 Front Panel of MDE/T-11 Cabinet

The MDE/T-11 bootstrap program displays the following message:

```
[CONNECTED TO HOST]
```

This message confirms that you are in the virtual terminal mode. If this message is not displayed, manually bootstrap the MDE/T-11 system as follows:

1. Halt the LSI-11/23 CPU by pressing the BREAK key on your terminal. The CPU halts and displays the console ODT prompt (@).

2. Bootstrap the LSI-11/23 system and place the MDE/T-11 system in virtual terminal mode by typing:

   ```
   @773000G<RET>
   ```

5.2 LOG ONTO OR BOOTSTRAP YOUR HOST SYSTEM

Before you can begin program development under VAX/VMS or RSX-11M, you must log onto the host. The log-in procedures for VAX/VMS and RSX-11M are given in Sections 5.2.1 and 5.2.2.

If you have an RT-11 host system, you bootstrap the system. Skip to Section 5.3 for the RT-11 bootstrapping procedure.

5.2.1 Logging onto a VAX/VMS Host

Log onto the VAX/VMS host system as follows:

1. Press the RETURN key to get the VAX/VMS user name and password prompts.

   ```
   <RET>
   ```

2. Type your account name and password:

   ```
   Username:  account-name<RET>
   Password:  password<RET>
   ```

VAX/VMS responds with the following message:

```
Welcome to VAX/VMS ...
```

5.2.2 Logging onto an RSX-11M Host

Log onto the RSX-11M host system as follows:

1. Respond to the RSX-11M prompt (>) by typing:

   ```
   >HELLO<RET>
   ```

2. Type your account or name and password:

   ```
   ACCOUNT OR NAME:  account-name<RET>
   PASSWORD:  password<RET>
   ```

RSX-11M responds with a message similar to the following:

```
RSX-11M BL26 MULTI-USER SYSTEM
GOOD AFTERNOON
19-MAY-82 LOGGED ON TERMINAL TT3:
```

5.3 BOOTSTRAP THE RT-11 HOST

Bootstrap the RT-11 host system as follows:

1. Assert a break to the host system on a VT100 or LA120 terminal by simultaneously holding down the CTRL key and pressing the slash (/) key.

   Refer to Section 5.1 for asserting a break on other terminals.

   Console ODT displays its prompt:

   ```
   @
   ```

2. Bootstrap RT-11XM on the host by typing:

   ```
   @773000G<RET>
   ```

   RT-11 then displays:

   ```
   RT-11XM V4.0
   .
   ```

   NOTE

   Messages sent by VAX/VMS to the MDE/T-11 console terminal, such as those from the system operator, are not displayed on the console screen. These messages are filtered out by the MDE/T-11 communications protocol.

5.4 CREATE A FILE FOR YOUR SOURCE PROGRAM

After logging onto or bootstrapping your host system, you must create a file for your source program. The program CALC demonstrates this step.

1. To create the source file for CALC, use one of the editors available under VAX/VMS, RSX-11M, or RT-11XM. If you are not familiar with these editors, refer to the VAX-11 EDT Editor Reference Manual or the PDP-11 Keypad Editor User's Guide.

2. After creating the file, type the appropriate source listing for CALC into the file.

Source listings for CALC are given below. The first listing applies to the VAX/VMS and RSX-11M host systems; the second applies to the RT-11 host system. The two listings for CALC differ only in the way code is positioned at absolute locations by the linker you use. (See Appendix D for a description of the techniques used to position code at absolute memory locations.)

5.4.1 CALC Source Listing for VAX/VMS or RSX-11M Host

```
        .TITLE CALC
        .IDENT /00.001/

        ; Program CALC accumulates a sum of the elements of the vector V in
        ; SUM1, and the sum of only those elements which are greater than or
        ; equal to 6 in SUM2.

              .PSECT ABS
              .=.+40000       ;Position code at relocatable address 40000

           V::  .WORD 5         ;Initialize the array
              .WORD 3
              .WORD 2
              .WORD 10
              .WORD 4
              .WORD 6
              .WORD 7
              .WORD 3
         SUM1::  .WORD
         SUM2::  .WORD

       START::  MOV #40000,SP   ;Initialize the stack pointer
              CLR SUM1          ;Initialize summation values
              CLR SUM2
              MOV #8.,R1        ;Initialize loop counter
              MOV #V,R0         ;Point to vector
           1$:  ADD (R0), SUM1  ;Add in current vector value
              MOV (R0)+, R2     ;Get a copy of current value
              JSR PC,QUANT
              DEC R1
              BGT 1$
              BR  START

        ; Subroutine to add a number to SUM2 if number is less than or equal to 6
        ; Call with number in R2

       QUANT::  CMP R2,#6       ;Less than 6?
              BGT 1$            ;No, do not add
              ADD R2,SUM2       ;Yes, add to SUM2
           1$:  RTS PC          ;Return to caller

              .END START        ;START is program transfer address
```

5.4.2 CALC Source Listing for RT-11 Host

```
        .TITLE CALC
        .IDENT /00.001/

        ; Program CALC accumulates a sum of the elements of the vector V in
        ; SUM1, and the sum of only those elements which are greater than or
        ; equal to 6 in SUM2.

             .ASECT
             .=40000

    V::      .WORD 5          ;Initialize the array
             .WORD 3
             .WORD 2
             .WORD 10
             .WORD 4
             .WORD 6
             .WORD 7
             .WORD 3
   SUM1::    .WORD
   SUM2::    .WORD

  START::    MOV SP           ;Initialize the stack pointer
             CLR SUM1         ;Initialize summation values
             CLR SUM2
             MOV #8.,R1       ;Initialize loop counter
             MOV #V,R0        ;Point to vector
     1$:     ADD (R0), SUM1   ;Add in current vector value
             MOV (R0)+, R2    ;Get a copy of current value
             JSR PC,QUANT
             DEC R1
             BGT 1$
             BR  START

        ; Subroutine to add a number to SUM2 if number is less than or equal to 6
        ; Call with number in R2

  QUANT::    CMP R2,#6        ;Less than 6?
             BGT 1$           ;No, do not add
             ADD R2,SUM2      ;Yes, add to SUM2
     1$:     RTS PC           ;Return to caller

             .END START       ;START is program transfer address
```

5.5 ASSEMBLE CALC WITH MACRO-11

After creating a source file for CALC, assemble CALC with the MACRO-11 assembler. To call the assembler, you use a different command line for each of the host operating systems, as discussed in the following subsections.

5.5.1 Assembling CALC on VAX/VMS Host

To assemble CALC and produce a listing under VAX/VMS, type the following command line:

```
$ MACRO/RSX/LIST CALC<RET>
```

Since the MACRO command assumes a default source file extension (.MAC), you do not have to include the file extension.

If CALC assembles successfully, VAX/VMS displays its command prompt ($).

You can now link CALC. Otherwise, look at your listing, correct the source file, and reassemble CALC.

5.5.2 Assembling CALC on RSX-11M Host

To assemble CALC and produce a listing under RSX-11M, type the following command line:

```
>MACRO CALC,CALC/-SP=CALC<RET>
```

Since the MACRO command assumes a default source file extension (.MAC), you do not have to include the file extension.

If CALC assembles successfully, RSX-11M displays its command prompt (>).

You can now link CALC. Otherwise, look at your listing, correct the CALC source file, and reassemble CALC.

5.5.3 Assembling CALC on RT-11XM Host

To assemble CALC and produce a listing under RT-11XM, type

```
.MACRO/LIST:CALC CALC<RET>
```

or

```
.R MACRO<RET>
*CALC,CALC=CALC<RET>
*<CTRL/C>
```

5.6 LINK CALC

After assembling CALC, you must link it. If you have a VAX/VMS or RSX-11M host, use TASK BUILDER (TKB). If you have a RT-11XM host, use LINK (RT-11 linker). The linking procedures you use under the different host operating systems are described in the following subsections.

5.6.1 Linking CALC on VAX/VMS or RSX-11M Host

Use TKB for linking CALC under VAX/VMS and RSX-11M. You use TKB with a VAX/VMS host because you operate in RSX-11M emulation mode during program development. When you link your program under a VAX/VMS or RSX-11M host, you must declare the following attributes to TKB so the resulting memory image (.EXE file under VAX/VMS, .TSK file under RSX-11M) can run under MDE/T-11.

- No RSX-11M task header
- An appropriate stack size
- A partition corresponding to physical memory in the MICRO/T-11 address space (the MICRO/T-11 application is an unmapped system)

To meet these requirements for the .EXE file or .TSK file:

1. Use the /-HD switch with TKB to create an image without a task header. The procedure for creating an image without a task header is described in the RSX-11M Task Builder Manual.

2. Use the TKB stack option to declare an appropriate stack size. You declare the maximum size of the stack to be added to the task image when your program runs in MICRO/T-11 memory. The stack option adds the number of words you specify to the beginning of the task image; the default value is 256(octal).

   The value you give for the stack size determines the physical location at which TKB begins positioning PSECTS. If you declare a stack size of Ø, TKB associates the relocatable addresses with physical addresses as required by the absolute code-positioning techniques described in Appendix D.

   Set the stack size to Ø as follows:

   ```
   TKB>STACK=Ø<RET>
   ```

3. Use the TKB partition option to install the task in a partition that corresponds to the physical memory of the MICRO/T-11 (Ø:160000). The partition option identifies the partition in memory for which the task is built. Since MICRO/T-11 applications are unmapped, the task is bound to physical memory. Therefore, you must install the task in a partition that corresponds to physical memory, as follows:

   ```
   TKB>PAR=DCT11:Ø:160000<RET>
   ```

Now you can link CALC with TKB.

To link CALC with TKB under VAX/VMS, type the following command line:

```
$MCR TKB<RET>
```

To link CALC with TKB under RSX-11M, type the following command line:

```
>TKB<RET>
```

For VAX/VMS or RSX-11M, TKB responds with the following prompt:

```
TKB>
```

At this point, link your modules with the following commands:

```
TKB>tskfile/-HD,mapfile/-SP, symbolfile=module1,module2,...<RET>
TKB>/<RET>
ENTER OPTIONS:
TKB>STACK=Ø<RET>
TKB>PAR=T11:Ø:160000<RET>
TKB>
```

where:

tskfile	= a memory image (.TSK) file to be loaded by MDE/T-11
mapfile	= a link map (.MAP) for your reference
symbolfile	= a symbol table (.STB) file to be loaded by MDE
module	= an object (.OBJ) file assembled by MACRO-11
/-HD	= a switch that produces a .TSK file without a header
/-SP	= a switch that prevents the .MAP file from being spooled to the line printer
STACK	= an option that specifies the maximum size of the stack to be added to the memory image
PAR	= an option that specifies the partition in memory for which the task is built

To return to the VAX/VMS or RSX-11M command level from TKB, type two slashes in response to the TKB prompt, as follows:

```
TKB>//<RET>
```

The host responds with $ (VAX/VMS) or > (RSX-11M).

In the following examples, CALC is used to demonstrate the linking procedure with TKB under VAX/VMS and RSX-11M.

Under VAX/VMS:

```
$MCR TKB<RET>
TKB>CALC/-HD,CALC/-SP,CALC=CALC<RET>
TKB>/<RET>
ENTER OPTIONS:
TKB>STACK=0<RET>
TKB>PAR=T11:0:160000<RET>
TKB>//<RET>
$
```

Under RSX-11M:

```
>TKB<RET>
TKB>CALC/-HD,CALC/-SP,CALC=CALC<RET>
TKB>/<RET>
ENTER OPTIONS:
TKB>STACK=0<RET>
TKB>PAR=T11:0:160000<RET>
TKB>//<RET>
>
```

NOTE

If you are using TKB Version 4.Ø or later, you can add the /IP qualifier after /-HD. Adding the /IP qualifier extends the upper boundary of the PARTITION option from 16ØØØØ to 177740. The linker can then locate code in this additional memory address space.

5.6.2 Linking CALC on RT-11 Host

In the following example, CALC is used to demonstrate the linking procedure with LINK under RT-11.

```
.LINK/LDA/SYMBOLTABLE CALC<RET>
```

NOTE

Refer to the RT-11 System User's Guide, Version 4.Ø, for further details on LINK.

Once you link CALC, you can down-line load, run, and debug the program, as described in Chapter 6.

CHAPTER 6

DEBUGGING

This chapter continues the description of the program development cycle. This chapter describes how to use MDE/T-11 commands, start MDE, load CALC, record the debugging session, and exit from MDE/T-11.

This chapter gives you an opportunity to use MDE/T-11 commands in a typical debugging session. Use CALC as described in the following sections, reading the explanations of the commands before using them.

To run CALC, you can use the memory simulator; you do not need target memory. However, if you decide to use target memory to run CALC, observe the precautions mentioned in Section 6.1.

To debug with MDE/T-11:

1. Start MDE/T-11 as described in Section 6.1.
2. Load the MDE/T-11 control software into LSI-11/23 memory from your host system.
3. Read the appropriate assembly listing of CALC.
4. Read the explanations of the commands before using them.
5. Type in the commands.

6.1 START MDE/T-11

Starting MDE/T-11 will vary slightly according to the host operating system you are using. If you have a VAX/VMS or RSX-11M host, begin with step 1. If you have an RT-11 host, skip to step 2.

1. Log onto or bootstrap your host by performing the steps listed in Sections 5.1 or 5.2.
2. Disconnect the pod plug from the target, if the plug is connected.

   If you use target memory in running CALC, connect the pod plug to the target, as shown in Figure 6-1.

CAUTION

Improperly inserting the pod plug into the target socket will damage the pod, pod plug, or both.

Carefully align PIN 1 on the pod plug with HOLE 1 in the target socket. The pod plug will slide easily into the corresponding holes in the target socket.

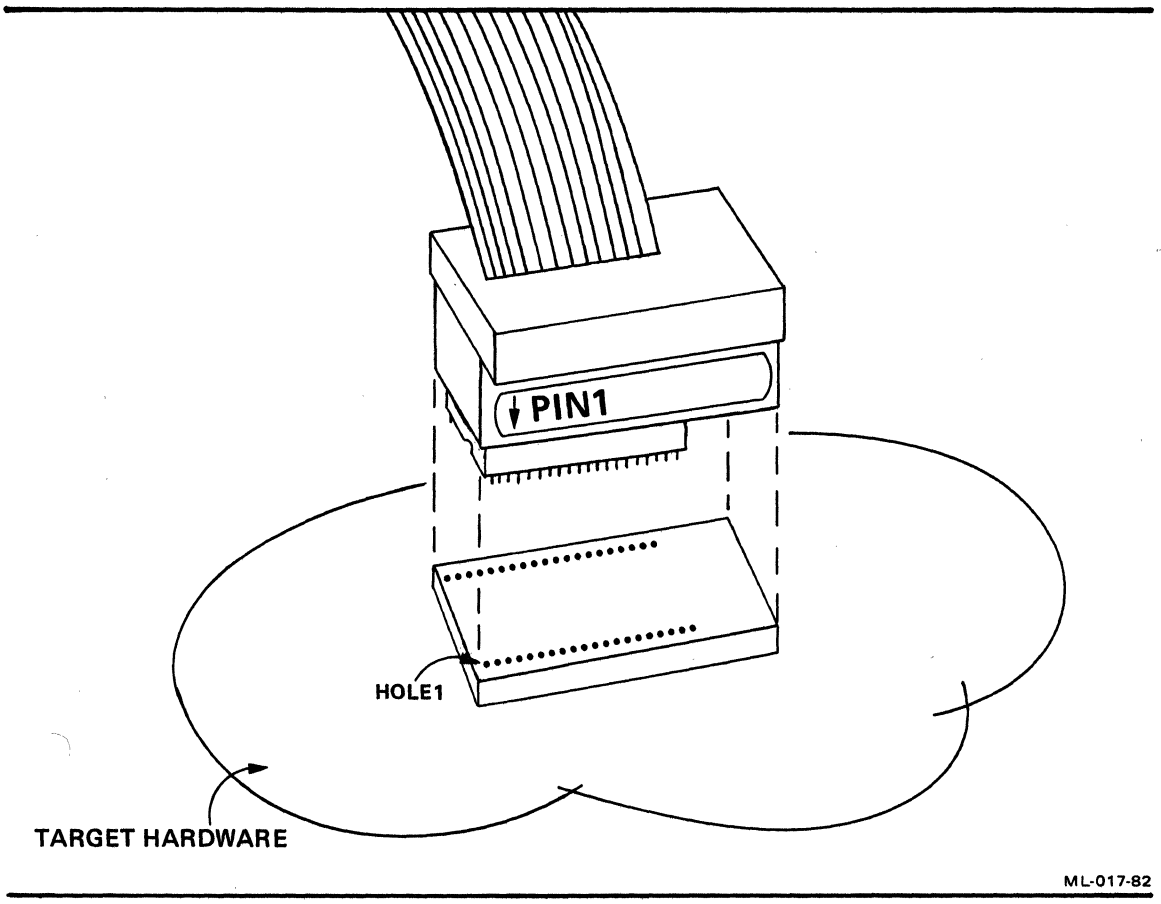


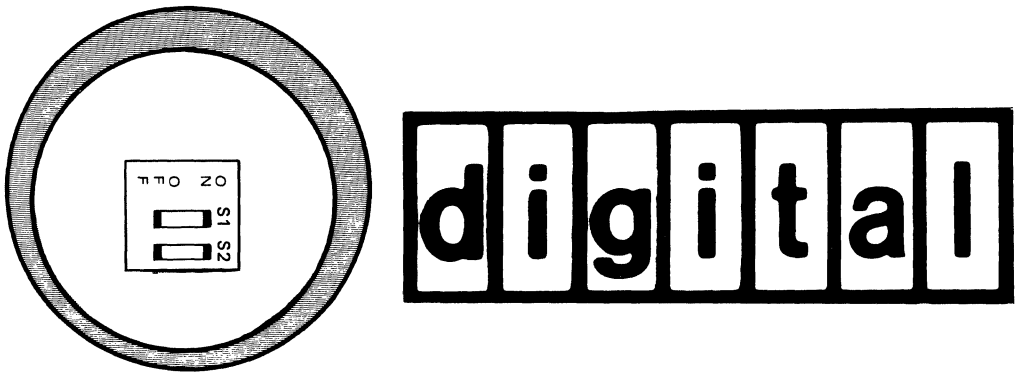
Figure 6-1 Connecting MDE/T-11 to the Target

3. If you are running CALC, you do not connect the pod to a target. in addition, you use the clock signal from the emulator. Therefore, turn on the clock switches, S1 and S2, in the pod (Figure 6-2).

   Use a small screwdriver to press the ON side of the switches down.

   With pod switches S1 and S2 on, you can use the emulator clock as the timing source for the MDE/T-11 system or an externally supplied clock (attached through a connector on the front panel of the MDE/T-11 cabinet).

   Refer to the description of the CONFIGURE CLOCK command in Chapter 8 for details on the use of the pod switches.



ML-018-82

Figure 6-2 Pod Clock Switches

4. Log into the account containing the MDE/T-11 software in its directory and type:

   VAX/VMS: $ RUN SYS$SYSTEM:MDET11<RET>
   RT-11XM: . R MDET11<RET>
   RSX-11M: > MDE<RET>

5. Down-line load the control software from the VAX/VMS host by typing L in response to the prompt from the host.

   Startup Option (Load,Restart,Diagnostic): L<RET>

   [Loading file "CSWB70.LDA", Please wait]

   At 9600 baud, down-line loading takes about 40 seconds. After the down-line load is complete, the MDE/T-11 response depends on whether you are using a VT100 or another type of terminal. If you are using a VT100, MDE/T-11 displays a header message and its prompt, as follows:

   MDE/T-11 V1.0
   MDE>

If your console terminal is a VT100 operating in the ANSI mode, the terminal is autosensed as such, and a scrolling region is set up in the bottom portion of the screen.

If you are using a VT100 to work with the example program, some of the information displayed by MDE/T-11 appears in the static display region.

6.2 RECORD YOUR DEBUGGING SESSION

MDE/T-11 lets you record all input and output of a debugging session. This logging consists of creating a file and placing in it commands you enter at the terminal and the MDE/T-11 software responses to those commands.

The commands for recording debugging sessions are:

```
MDE>SET LOG CALC.LOG<RET>
MDE>SET OUTPUT LOG<RET>
```

To specify the name of the log file and to turn the logging mechanism on, use the commands SET LOG CALC.LOG and SET OUTPUT LOG. When you execute these commands, your debugging session (from point of execution onward) is recorded in the file CALC.LOG.

```
MDE>SHOW OUTPUT<RET>
OUTPUT set to:  TERMINAL, VERIFY, LOG (Log file is 'CALC.LOG')
```

To display the output settings in effect, use the SHOW OUTPUT command. The display produced by this command includes the name of your log file and the logging status (on or off).

6.3 SET UP THE TARGET

Before debugging a target with MDE/T-11, you describe your target to the MDE/T-11 software. The commands in the the example below describe the following characteristics of your target:

- The contents of the MICRO/T-11 mode register and the source from which the mode register is to be read
- The mapping and protection of memory in the MICRO/T-11 address space

The two commands for setting up the target are shown in following examples:

```
MDE>CONFIGURE MODE EMULATOR BIT16 STATIC<RET>
```

The CONFIGURE MODE command tells the MDE/T-11 debugger to read the MICRO/T-11 mode register from the emulator. The BIT16 and STATIC parameters specify that the target has a bus width of 16 bits and static memory. Other mode register settings are determined by the default parameters in the CONFIGURE MODE command. (See the description of the CONFIGURE MODE command in Chapter 8 for these defaults.)

```
MDE>CONFIGURE MEMORY 36000:50000 SIMULATOR<RET>
```

The CONFIGURE MEMORY command tells the debugger to simulate target memory in the memory simulator from location 36000 to location 50000.

```
MDE>SHOW CONFIGURE<RET>
Mode Register:    Read from EMULATOR
Mode settings:    NORMAL, STANDARD, STATIC, PROCESSOR, USER,
                  Start Addr = 140000, Bus Width  = 16 bits

Processor Clock: EMULATOR
Fetch Timeout:   DISABLED
State Analyzer:  Clock A+; AI = INTERRUPT; Trace AFTER /Retain:NA
```

```
Memory Map (octal):

 From        To       Where     Write Prot

------     ------    ------     ----------
000000     035777    Absent
036000     050377    Sim              No
050400     177777    Absent
MDE>
```

The SHOW CONFIGURE command displays the current configuration of the target. In the preceding example, the display shows the:

- Source from which the MICRO/T-11 mode register is read (in this case, emulator as opposed to target)
- Settings you previously specified for the MICRO/T-11 mode register
- Source of the processor clock (default shown)
- Status of the fetch timeout feature
- Configuration of the state analyzer (defaults shown)
- Mapping, location, and protection of MICRO/T-11 memory

```
MDE>SHOW MODE<RET>
     The current modes are:  OCTAL,INSTRUCTION,SYMBOLIC
```

The SHOW MODE command shows the display modes in effect. In the example above, the debugger's response to the SHOW MODE command indicates that the default display modes (OCTAL, INSTRUCTION, and SYMBOLIC) are set.

In the OCTAL radix mode, MDE/T-11 interprets and displays numbers in base 8. The INSTRUCTION display mode displays data in MACRO-11 instruction format. In the SYMBOLIC address mode, MDE/T-11 displays addresses in a symbol+offset format.

```
MDE>POWER UP<RET>
```

The POWER UP command specifies the source of the power-up signal and sends the power-up signal to the MICRO/T-11 microprocessor. You can use the CONFIGURE MODE command only when power is down. You must therefore follow any CONFIGURE MODE command with a POWER UP command. In the preceding example, the power-up signal comes from the emulator.

After you have executed all the commands in the preceding example, you can load and run CALC as described in the next section.

6.4 LOAD CALC

Two assembly listings are presented below. The first one is for use with a VAX/VMS or an RSX-11M host; the second, with an RT-11 host.

You can determine from the first listing the memory locations of the padding or other irrelevant data in the memory image (.TSK file) produced by TKB. Use this information to exclude memory locations or ranges of locations containing irrelevant data from the .TSK file.

NOTE

If you have an RT-11 host, skip to Section 6.4.2 for a CALC listing assembled under an RT-11 host operating system.

6.4.1 Assembly Listings

1. For VAX/VMS or RSX-11M host:

```
CALC    MACRO M1114  26-MAR-82 11:30  PAGE 1


      1                                         .TITLE CALC
      2
      3                                 ; PROGRAM CALC COMPUTES THE SUM OF THE ELEMENTS
      4                                 ; OF A VECTOR V IN SUM1 AND THE SUM OF THE
      5                                 ; ELEMENTS OF V THAT ARE LESS THAN OR
                                        ; EQUAL TO  6 IN SUM2.
      6
      7                                         .IDENT /00.001/

      8 000000                                  .PSECT ABS
      9          040000'                        .=.+40000       ;POSITION CODE AT RELOCATABLE ADDRESS 40000

     10 040000  000005                  V::     .WORD 5         ;INITIALIZE THE ARRAY
     11 040002  000003                          .WORD 3
     12 040004  000002                          .WORD 2
     13 040006  000010                          .WORD 10
     14 040010  000004                          .WORD 4
     15 040012  000006                          .WORD 6
     16 040014  000007                          .WORD 7
     17 040016  000003                          .WORD 3
     18 040020  000000                  SUM1::  .WORD
     19 040022  000000                  SUM2::  .WORD

     20 040024  012706  040000          START:: MOV #40000,SP   ;INITIALIZE THE STACK POINTER
     21 040030  005067  177764                  CLR SUM1        ;INITIALIZE SUMMATION VALUES
     22 040034  005067  177762                  CLR SUM2
     23 040040  012701  000010                  MOV #8.,R1      ;INITIALIZE LOOP COUNTER
     24 040044  012700  040000'                 MOV #V,R0       ;POINT TO VECTOR
     25 040050  061067  177744          1$:     ADD (R0), SUM1  ;ADD IN CURRENT VECTOR VALUE
     26 040054  012002                          MOV (R0)+, R2   ;GET A COPY OF CURRENT VALUE
     27 040056  004767  000006                  JSR PC,QUANT
     28 040062  005301                          DEC R1
     29 040064  003371                          BGT 1$
     30 040066  000756                          BR  START
     31
     32                                 ; SUBROUTINE TO ADD A NUMBER TO SUM2 IF NUMBER IS
                                        ; LESS THAN OR EQUAL TO 6
     33
     34 040070  020227  000006          QUANT:: CMP R2,#6       ;LESS THAN OR
                                                ;   EQUAL TO 6?
     35 040074  003002                          BGT 1$          ;NO, DO NOT ADD
     36 040076  060267  177720                  ADD R2,SUM2     ;YES, ADD TO SUM2
     37 040102  000207                  1$:     RTS PC          ;RETURN TO CALLER
     38
     39          040024'                        .END START      ;START IS PROGRAM TRANSFER ADDRESS

CALC    MACRO M1114  26-MAR-82 11:30  PAGE 1-1
SYMBOL TABLE

QUANT   040070RG    002 START   040024RG    002 SUM1    040020RG    002 SUM2    040022RG    002 V    040000RG    002


  ABS.  000000      000
        000000      001
ABS     040104      002
ERRORS DETECTED:  0
VIRTUAL MEMORY USED:  40 WORDS  ( 1 PAGES)
DYNAMIC MEMORY:  9224 WORDS  ( 35 PAGES)
ELAPSED TIME:  00:00:05
CALC,CALC/-SP=CALC.MAC
```

2. For RT-11 host:

```
CALC    MACRO V04.00  26-MAY-82 10:44:12 PAGE 1

      1                                         .TITLE CALC
      2                                         .IDENT /00.001/
      3
      4                                 ; PROGRAM CALC ACCUMULATES A SUM OF THE ELEMENTS OF THE VECTOR V IN
      5                                 ; SUM1, AND THE SUM OF ONLY THOSE ELEMENTS WHICH ARE GREATER THAN OR
      6                                 ; EQUAL TO 6 IN SUM2.
      7
      8 000000                                  .ASECT
      9         040000                          .=40000
     10
     11 040000  000005                  V::     .WORD   5               ;INITIALIZE THE ARRAY
     12 040002  000003                          .WORD   3
     13 040004  000002                          .WORD   2
     14 040006  000010                          .WORD   10
     15 040010  000004                          .WORD   4
     16 040012  000006                          .WORD   6
     17 040014  000007                          .WORD   7
     18 040016  000003                          .WORD   3
     19 040020  000000                  SUM1::  .WORD   0
     20 040022  000000                  SUM2::  .WORD   0
     21
     22 040024  012706  040000          START:: MOV     #40000,SP       ;INITIALIZE THE STACK POINTER
     23 040030  005067  177764                  CLR     SUM1            ;INITIALIZE SUMMATION VALUES
     24 040034  005067  177762                  CLR     SUM2
     25 040040  012701  000010                  MOV     #8.,R1          ;INITIALIZE LOOP COUNTER
     26 040044  012700  040000                  MOV     #V,R0           ;POINT TO VECTOR
     27 040050  061067  177744          1$:     ADD     (R0), SUM1      ;ADD IN CURRENT VECTOR VALUE
     28 040054  012002                          MOV     (R0)+, R2       ;GET A COPY OF CURRENT VALUE
     29 040056  004767  000006                  JSR     PC,QUANT
     30 040062  005301                          DEC     R1
     31 040064  003371                          BGT     1$
     32 040066  000756                          BR      START
     33
     34                                 ; SUBROUTINE TO ADD A NUMBER TO SUM2 IF NUMBER IS LESS THAN OR EQUAL TO 6
     35                                 ; CALL WITH NUMBER IN R2
     36
     37 040070  020227  000006          QUANT:: CMP     R2,#6           ;LESS THAN 6?
     38 040074  003002                          BGT     1$              ;NO, DO NOT ADD
     39 040076  060267  177720                  ADD     R2,SUM2         ;YES, ADD TO SUM2
     40 040102  000207                  1$:     RTS     PC              ;RETURN TO CALLER
     41
     42         040024                          .END    START           ;START IS PROGRAM TRANSFER ADDRESS

CALC    MACRO V04.00  26-MAY-82 10:44:12 PAGE 1-1
SYMBOL TABLE

QUANT   040070 G        START   040024 G        SUM1    040020 G        SUM2    040022 G        V       040000 G


. ABS.  040104    000
        000000    001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8192 WORDS  ( 32 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  61 PAGES
MD:CALC,MD:CALC=MD:CALC
```

6.4.2 Applicable Commands

The commands to load CALC are:

```
MDE>LOAD/CLEAR/EXCLUDE:(0:37777,50400:160000) CALC<RET>
;MDE-I-LOAINPROG, Binary load in progress, please wait...
;MDE-I-LOAINPROG, Symbol load in progress, please wait...
;MDE-I-LOAINPROG, 5 symbols loaded for CALC  00.001
;MDE-S-LOAGOODLD, Load complete
```

To load CALC into simulator memory, use the LOAD command with the /CLEAR and /EXCLUDE qualifiers.

The /CLEAR qualifier clears the MDE/T-11 internal symbol table and speeds up the loading process by eliminating the need for MDE/T-11 to check for duplicate entries in its symbol table.

The /EXCLUDE qualifier excludes address ranges containing irrelevant portions of the .TSK file such as padding. When you use this qualifier, you also speed up the loading process, as irrelevant data in the load has been eliminated.

NOTE

If you have linked CALC using LINK (RT-11 linker), you do not need to use the /EXCLUDE qualifier with the LOAD command, because files in LDA format include only code or data produced during assembly.

When you execute the LOAD command as shown above, CALC.EXE is loaded into the MICRO/T-11 address space, program symbols are loaded from the file CALC.STB into the debugger symbol table, and the transfer address is loaded into the program counter (PC). If you have a VT100 console terminal, the progress of the load is displayed in the left-hand corner of the static region.

When you execute LOAD, a number of messages (shown above) are displayed. The first two messages tell you that your program code and your program symbols are being loaded. (If you have a VT100 terminal, this information is presented as relative block numbers in the static region.) The next message tells you how many program symbols were loaded. The last message tells you that loading is complete.

```
MDE>SHOW SYMBOL/ALL<RET>

     Symbol                    Value
     QUANT                     40070
     START                     40024
     V                         40000
     SUM1                      40020
     SUM2                      40022
```

The SHOW SYMBOL command with the /ALL qualifier displays the CALC program symbols.

6.5 EXECUTE YOUR PROGRAM

Now execute the program.

Here are the commands you use to execute CALC:

```
MDE>GO<RET>
;MDE-I-PROEXESTA, Processor execution started at PC:START
```

The GO command starts program execution at the current PC (in this case, START). As an option, you can supply a starting address.

```
MDE>HALT<RET>
;MDE-I-PROHALAT, Processor execution halted at PC:START+4: CLR SUM1
     R0:  40020
     R1:      0
     R2:      3
     R3:      0
     R4:      0
     R5:      0
     SP:  40000
     PC:  START+4
     PS:    340
```

The HALT command stops program execution. When you execute the HALT command, MICRO/T-11 registers are read and updated in the VT100 display. If you are not using a VT100, the registers are displayed serially (register contents may be different from those shown in this example). In this example, address may be any instruction boundary in the program; see either assembly listing of CALC.

```
MDE>STEP<RET>
;MDE-I-PROSTETO, Processor single stepped to PC:START+10: CLR SUM2
     R0:  40020
     R1:      0
     R2:      3
     R3:      0
     R4:      0
     R5:      0
     SP:  40000
     PC: START+10
     PS:    344
```

The STEP command executes one or more program instructions at a time. As in the HALT command, MICRO/T-11 registers are displayed. If you are using a VT100 as your console terminal, the registers are displayed in the static region. If you are using another type of terminal, such as a VT52 or LA36, the registers are displayed serially, as shown above.

6.6 EXAMINE AND CHANGE MEMORY AND REGISTERS

The EXAMINE and DEPOSIT commands examine and change memory. In the following example, the EXAMINE command determines the contents of all memory locations between the locations symbolized by START and QUANT+12.

```
MDE>EXAMINE START:QUANT+12<RET>
        START : MOV        #V,SP
      START+4 : CLR        SUM1
     START+10 : CLR        SUM2
     START+14 : MOV        #10,R1
     START+20 : MOV        #V,R0
     START+24 : ADD        @R0,SUM1
     START+30 : MOV        (R0)+,R2
     START+32 : JSR        PC,QUANT
     START+36 : DEC        R1
     START+40 : BGT        START+24
     START+42 : BR         START
        QUANT : CMP        R2,#6
      QUANT+4 : BGT        QUANT+12
     QUANT+12 : ADD        R2,SUM2
     QUANT+12 : RTS        PC
```

The example above illustrates the disassembly feature of MDE/T-11. When you execute the EXAMINE command, all instructions and arguments within the range of locations specified in your command line are displayed.

```
MDE>EXAMINE/WORD SUM1:SUM2<RET>
            SUM1 :  0
            SUM2 : 27
```

To display memory in the current radix, use the EXAMINE command with the /WORD qualifier. In the example above, EXAMINE/WORD is used to examine the memory locations containing the results or partial results of CALC.

```
MDE>DEPOSIT SUM1:SUM2 = Ø<RET>

MDE>DEPOSIT RØ:R5 = Ø,Ø,Ø,Ø,Ø,Ø<RET>
```

The DEPOSIT command changes the contents of memory locations or registers or ranges of memory locations and registers. The command initializes the variables SUM1 and SUM2. You initialize registers as shown in the following example of the DEPOSIT command.

```
MDE>EXAMINE PC<RET>
     PC:  START+1Ø
```

The EXAMINE command can also be used to display MICRO/T-11 registers. If you are using a VT1ØØ, the specified register or registers in the command are shown in reverse video in the static display region. On the terminals, the registers are listed serially.

```
MDE>DEPOSIT 4ØØ66 = "BR ."<RET>
```

In the example above, the DEPOSIT command deposits a PDP-11 instruction into a memory location. Assembled by the MDE/T-11 line assembler, this instruction replaces the BR START instruction at location 4ØØ66. CALC now executes only once and goes into an infinite loop caused by the "BR ." instruction.

6.7 SET AND CANCEL EVENTS

You can set both predefined and user-defined events (UDEs) with MDE/T-11. The predefined events you can set are breakpoints, watchpoints, and tracepoints.

A UDE is a general type of event that detects a specific pattern in MICRO/T-11 bus signals. UDEs are described in Chapters 3 and 8.

You can use various commands to set and cancel events.

```
MDE>SET BREAK START+1Ø<RET>
```

You use the SET BREAK command to halt execution of the program when the MICRO/T-11 fetches an instruction at an address specified in your command line. In the preceding example, program execution halts after the variables SUM1 and SUM2 are initialized.

```
MDE>SHOW BREAK A:Ø<RET>
#Ø  A:Ø Breakpoint /COUNT:1 /AFTER:1 (1 left)  Address = START+1Ø
```

The SHOW BREAK command displays the:

- Identification number of the event (event ordinal)
- Comparator being used to set the event
- Value of the /COUNT qualifier
- Value of the /AFTER qualifier

- Number of counts remaining
- Address at which the event is set (address expression)

```
MDE>SET WATCH SUM2<RET>
```

The SET WATCH command halts program execution when the MICRO/T-11 writes to a memory location. In the example above, the SET WATCH command is used to monitor the value in location SUM2.

```
MDE>SHOW WATCH<RET>
#1  A:1 Watchpoint /COUNT:1 /AFTER:1 (1 left)  Address = SUM2
```

The SHOW WATCH command displays all watchpoints that have been set.

```
MDE>SET TRACE QUANT<RET>
```

The SET TRACE command traces instruction fetches at a given address. When a tracepoint occurs, program execution is halted temporarily, a message is displayed at the console, and program execution is resumed. In this example, a tracepoint occurs whenever the subroutine QUANT is entered.

```
MDE>SHOW ALL<RET>
#Ø  A:Ø Breakpoint /COUNT:1 /AFTER:1 (1 left)  Address = START+1Ø
#1  A:1 Watchpoint /COUNT:1 /AFTER:1 (1 left)  Address = SUM2
#2  A:2 Tracepoint /COUNT:1 /AFTER:1 (1 left)  Address = QUANT
```

To display the events set thus far in the debugging session, use the SHOW command with the /ALL parameter.

```
MDE>GO START<RET>
;MDE-I-PROEXESTA, Processor execution started at PC:START+14

BREAK at PC = START+1Ø
```

The GO START command starts the execution of CALC at the location symbolized by START. When your program encounters the breakpoint set at location START+1Ø, it halts.

```
MDE>CANCEL BREAK A:Ø<RET>
MDE>GO<RET>
;MDE-I-PROEXESTA, Processor execution started at PC:START+14

TRACE at PC = QUANT

WATCH at SUM2; value was: Ø, is now: 5; current PC = QUANT+12
```

To cancel the breakpoint set in comparator Ø on state analyzer A, use the CANCEL BREAK command. Use the GO command to resume program execution at the current PC. When you issue the GO command, the instruction at QUANT is executed and the tracepoint is reported. In addition, the contents of SUM2 are changed and the watchpoint is reported. After the watchpoint is reported, CALC halts.

```
CANCEL WATCH A:1<RET>
```

To cancel the watchpoint at SUM2 in this example, use the CANCEL WATCH command.

```
MDE>GO<RET>
;MDE-I-PROEXESTA, Processor execution started at PC:QUANT+12

TRACE at PC = QUANT

TRACE at PC = QUANT

TRACE at PC = QUANT

TRACE at PC = QUANT

TRACE at PC = QUANT

TRACE at PC = QUANT

TRACE at PC = QUANT

HALT<RET>
;MDE-I-PROHALAT, Processor execution halted at PC:START+42: BR START
R0:  40020
R1:      0
R2:      3
R3:      0
R4:      0
R5:      0
SP:  40000
PC: START+42
PS:    344

MDE>GO<RET>
;MDE-I-PROEXESTA, Processor execution started at PC:START+42
```

Resume program execution with the GO command.

6.8 TRACE BUS CYCLE

You use the CLEAR TRACERAM to initialize the trace buffer; the CLEAR TRACERAM command does an implicit reset of the trace mode and the retain count. You specify the conditions of tracing with the CONFIGURE ANALYZER command. (See the description of CLEAR TRACERAM and CONFIGURE ANALYZER in Chapter 8 for additional information.)

In the following example, the trace RAM is first cleared of previous bus cycle trace data and then configured to capture bus cycle data continuously. After the trace RAM has been configured, CALC is executed.

```
MDE>CLEAR TRACERAM<RET>
```

To initialize the 1024 locations in the trace RAM, use the CLEAR TRACERAM command.

```
MDE>CONFIGURE ANALYZER TRACE ALWAYS<RET>
```

You use the CONFIGURE ANALYZER command to tell MDE/T-11 to turn on the trace RAM throughout the execution of CALC.

```
MDE>HALT<RET>
;MDE-I-PROHALAT, Processor execution halted at PC:START+42: BR START
     R0:  40020
     R1:      0
     R2:      3
     R3:      0
     R4:      0
     R5:      0
     SP:  40000
     PC: START+42
     PS:    344

MDE>DISPLAY TRACERAM 5 6<RET>

Frame  Bus Trans                                  SEL        AI       Probe
  No.  Status          Address      Data  Lines     Lines    Lines A
-----  ---------  -----------  -------  -----  --------  --------
    5  Fetch         START+42      777     01  11111111  00000000
    4  Fetch         START+42      777     01  11111111  00000000
    3  Fetch         START+42      777     01  11111111  00000000
    2  Fetch         START+42      777     01  11111111  00000000
    1  Fetch         START+42      777     01  11111111  00000000
    0  Fetch         START+42      777     01  11111111  00000000
```

When CALC is stopped by the HALT command at START+42, the trace RAM contains the last 1024 cycles of the MICRO/T-11; the 6 most recent cycles are displayed by the DISPLAY TRACERAM command. In the example above, the trace RAM display shows that the MICRO/T-11 is executing the "BR ." instruction at START+42.

6.9 END THE DEBUGGING SESSION

After completing your debugging session, you may want a listing of the session. To obtain this listing, turn off the logging mechanism with the SET OUTPUT NOLOG command and use the appropriate command to print the listing (PRINT with VAX/VMS, QUE with RSX, PRINT with RT-11) when you return to the host operating system. To return to the host operating system, use the MDE/T-11 EXIT command.

Here are the commands you use in ending a debugging session.

```
MDE>SET OUTPUT NOLOG<RET>
```

To turn off the logging mechanism, use the SET OUTPUT NOLOG command. You can later print out the log file when you return to DCL command level on the host.

```
MDE>EXIT<RET>

[Comm stats (tot/tmo/cs/sn):  856/1/0/1]

[CONNECTED TO HOST]
```

The log file (CALC.LOG) for the debugging session you have just completed is shown below.

```
SHOW OUTPUT
!Output set to: TERMINAL, VERIFY, LOG (Log file is 'CALC.LOG')
CONFIGURE MODE EMULATOR BIT16 STATIC
CONFIGURE MEMORY 36000:50000 SIMULATOR
SHOW CONFIGURE
!Mode Register:   Read from EMULATOR
!Mode settings:   NORMAL, STANDARD, STATIC, PROCESSOR, USER,
!                 Start Addr = 140000, Bus Width = 16 bits
!
!Processor Clock: EMULATOR
!Fetch Timeout:   DISABLED
!State Analyzer:  Clock  A+;  AI = INTERRUPT;  Trace AFTER  /Retain: NA
!
!Memory Map (octal):
!
! From      To      Where   Write Prot
!------   ------   ------   ----------
!000000   035777   Absent
!036000   050377   Sim            No
!050400   177777   Absent
SHOW MODE
!The current modes are: OCTAL,INSTRUCTION,SYMBOLIC
POWER UP
LOAD/CLEAR/EXCLUDE:(0:37777,50400:160000) CALC
!;MDE-I-LOASYMLOA, 5 symbols loaded for CALC   00.001
!;MDE-S-LOAGOODLD, Load complete
SHOW SYMBOL/ALL
!       Symbol                  Value
!       QUANT                   40070
!       START                   40024
!       V                       40000
!       SUM1                    40020
!       SUM2                    40022
GO
!;MDE-I-PROEXESTA, Processor execution started at PC:START
HALT
!;MDE-I-PROHALAT, Processor execution halted at PC:START+4: CLR SUM1
!R0:  40020
!R1:      0
!R2:      3
!R3:      0
!R4:      0
!R5:      0
!SP:  40000
!PC: START+4
!PS:    340
STEP
!;MDE-I-PROSTETO, Processor single stepped to PC:START+10: CLR SUM2
!R0:  40020
!R1:      0
!R2:      3
!R3:      0
!R4:      0
!R5:      0
!SP:  40000
!PC: START+10
!PS:    344
EXAMINE START:QUANT+12
!         START : MOV      #V,SP
!       START+4 : CLR      SUM1
!      START+10 : CLR      SUM2
!      START+14 : MOV      #10,R1
!      START+20 : MOV      #V,R0
!      START+24 : ADD      @R0,SUM1
!      START+30 : MOV      (R0)+,R2
```

```
!       START+32 : JSR       PC,QUANT
!       START+36 : DEC       R1
!       START+40 : BGT       START+24
!       START+42 : BR        START
!          QUANT : CMP       R2,#6
!        QUANT+4 : BGT       QUANT+12
!        QUANT+6 : ADD       R2,SUM2
!       QUANT+12 : RTS       PC
EXAMINE/WORD SUM1:SUM2
!           SUM1 : 0
!           SUM2 : 27
DEPOSIT SUM1:SUM2 = 0
DEPOSIT R0:R5 = 0,0,0,0,0,0
EXAMINE PC
!PC: START+10
DEPOSIT 40066 = "BR ."
SET BREAK START+10
SHOW BREAK A:0
!#0  A:0 Breakpoint /COUNT:1 /AFTER:1 (1 left)  Address = START+10
SET WATCH SUM2
SHOW WATCH
!#1  A:1 Watchpoint /COUNT:1 /AFTER:1 (1 left)  Address = SUM2
SET TRACE QUANT
SHOW ALL
!#0  A:0 Breakpoint /COUNT:1 /AFTER:1 (1 left)  Address = START+10
!#1  A:1 Watchpoint /COUNT:1 /AFTER:1 (1 left)  Address = SUM2
!#2  A:2 Tracepoint /COUNT:1 /AFTER:1 (1 left)  Address = QUANT
GO START
!;MDE-I-PROEXESTA, Processor execution started at PC:START
!
!BREAK at PC = START+10
CANCEL BREAK A:0
GO
!;MDE-I-PROEXESTA, Processor execution started at PC:START+14
!
!TRACE at PC = QUANT
!
!WATCH at SUM2; value was: 0, is now: 5; current PC = QUANT+12
CANCEL WATCH A:1
GO
!;MDE-I-PROEXESTA, Processor execution started at PC:QUANT+12
!
!TRACE at PC = QUANT
!
!TRACE at PC = QUANT
!
!TRACE at PC = QUANT
!
!TRACE at PC = QUANT
!
!TRACE at PC = QUANT
!
!TRACE at PC = QUANT
!
!TRACE at PC = QUANT
HALT
!;MDE-I-PROHALAT, Processor execution halted at PC:START+42: BR START
!R0:  40020
!R1:      0
!R2:      3
!R3:      0
!R4:      0
!R5:      0
!SP:  40000
!PC: START+42
```

```
!PS:    344
GO
!;MDE-I-PROEXESTA, Processor execution started at PC:START+42
CLEAR TRACERAM
CONFIGURE ANALYZER TRACE ALWAYS
HALT
!;MDE-I-PROHALAT, Processor execution halted at PC:START+42: BR START
!R0:  40020
!R1:      0
!R2:      3
!R3:      0
!R4:      0
!R5:      0
!SP:  40000
!PC: START+42
!PS:    344
DISPLAY TRACERAM 5 6
!Frame  Bus Trans                                    SEL        AI       Probe
!  No.  Status          Address       Data  Lines     Lines    Lines A
!-----  ---------  -----------   -------  -----  --------  --------
!     5  Fetch         START+42       777     01  11111111  00000000
!     4  Fetch         START+42       777     01  11111111  00000000
!     3  Fetch         START+42       777     01  11111111  00000000
!     2  Fetch         START+42       777     01  11111111  00000000
!     1  Fetch         START+42       777     01  11111111  00000000
!     0  Fetch         START+42       777     01  11111111  00000000
SET OUTPUT NOLOG
```

CHAPTER 7

COMMAND LANGUAGE

This chapter describes the major elements in the MDE/T-11 command language.

7.1 EXPRESSIONS

Most numerical values used with MDE/T-11 commands can be entered or displayed as expressions. You arithmetically combine signed or unsigned symbols or numbers and special characters to form expressions. A register, however, cannot be part of an expression.

Some examples of expressions

```
DOG    CAT+14    -BLUE+@HAT    VALUE-(LAST-(MID-2))
```

The results of expressions evaluated by MDE/T-11 are truncated to 16 bits.

7.1.1 Numbers

Numbers can be entered or displayed in any of four radixes.

Radix	Character Set
Binary	Ø and 1
Octal	Ø through 7
Decimal	Ø through 9
Hexadecimal	Ø through 9 and A through F

A hexadecimal number that begins with a letter must be entered with a leading zero. For example:

```
ØA1
```

Some examples of numbers:

```
1Ø   Ø  777   ØFF  1A2
```

MDE/T-11 uses 16-bit arithmetic in decoding numbers from your commands and in displaying numbers. In addition, overflow occurring during number decoding or arithmetic operations is truncated without warning.

The following examples illustrate expressions you might use and the numerical results obtained from these expressions.

Radix Mode	You Type	Resulting Value	
		Octal	Decimal
8	777777	177777	-1
8	400000	0	0
10	65535 + 1	0	0
8	100000	100000	-32768

You can use the following temporary radix control operators when entering numbers that are not in the current radix mode (Section 7.2.1):

%D'n' A decimal number

%O'n' An octal number

%X'n' A hexadecimal number

%B'n' A binary number

%R'xxx' From 0 to 3 Radix-50 characters

7.1.2 Symbols

You can use the following symbols.

Your program's global symbols

Symbols you create with MDE/T-11

MDE/T-11 can use your program's global symbols (but not its local symbols), provided a symbol table file (.STB) is produced when the program is linked. In addition, you can create new symbols by using the DEFINE command.

A symbol can be up to 30 characters long, and can contain any combination of letters, digits, dollar signs or underscore characters, but the first character cannot be a digit.

Some examples of symbols:

A1 $1 ALPHA$BETA ENTRY_1 TYPE_NO

7.1.3 Special Characters

A period (.) represents the current address, and a backslash (\) represents the previous address.

7.1.3.1 **Current Address Indicator** - The current address indicator (.) specifies the address last used by MDE/T-11 in an EXAMINE or DEPOSIT command. For example, if you want to look at a location in more than one radix, you can first examine the address by typing the following:

```
EXAMINE ENTRY1+26
```

In response to this command, MDE/T-11 prints the contents of that address in the current mode (for example, octal). To look at the same location in decimal, you can type the following:

```
EXAMINE/DECIMAL .
```

The period specifies the current address (ENTRY1+26).

7.1.3.2 **Previous Address Indicator** - The previous address indicator (\) specifies the previous value of the current address indicator (.). For example, consider the following sequence of addresses:

```
EXAMINE 1000
EXAMINE 2000
```

If you want to reexamine location 1000, you can type the following:

```
EXAMINE \
```

This displays the contents of the previous address (1000).

7.1.4 Operators

The arithmetic operators are as follows:

+ (addition)

- (subtraction)

* (multiplication)

/ (division)

All results of arithmetic are truncated to 16 bits.

The fetch operator is an at-sign (@); you can also use a period (.) as a fetch operator.

The fetch operator lets you specify the contents of memory locations in expressions (in addition to addresses). For example, suppose you have the following locations.

Symbol	Value	Contents
ALPHA	36020	3
BETA	36022	5
GAMMA	36024	36022

The EXAMINE command displays the contents.

```
MDE>EXAMINE ALPHA<RET>
ALPHA:     3
```

The SHOW SYMBOL command gets the value.

```
MDE>SHOW SYMBOL ALPHA<RET>

     Symbol     Value

     ALPHA      36Ø2Ø
```

And to change the contents of ALPHA to the contents of BETA, you need a fetch operator for BETA.

```
MDE>DEPOSIT ALPHA=@BETA<RET>
```

This command changes the contents of ALPHA to 5.

7.1.5 Order of Evaluation

MDE/T-11 evaluates expressions according to the following priorities:

First: signed numbers, signed expressions, and @ operations

Second: * and / operations

Third: + and - operations

If successive operators have the same priority, evaluation is from left to right.

You can override the normal order of evaluation by selectively enclosing operations in parentheses. For example, consider the following expression:

```
@V + 6
```

Since the fetch operator has highest priority, the expression is evaluated as if parentheses were inserted.

```
(@V) + 6
```

This expression resolves to the value of the contents of location V incremented by 6. To specify the contents of location V + 6, you must insert parentheses.

```
@(V + 6)
```

The EVALUATE command evaluates expressions and can be used to check the value of expressions before use.

7.2 MODES

The mode setting establishes the form in which MDE/T-11 interprets or displays numerical values when more than one form is possible. Three settings are always in effect, one for each of the following modes.

Mode	Settings			
Radix	BINARY	OCTAL	DECIMAL	HEXADECIMAL
Display	WORD	BYTE	INSTRUCTION	ASCII RAD5Ø
Address	SYMBOLIC	NOSYMBOLIC		

7.2.1 Radix Mode

The radix mode affects the way numbers are interpreted by MDE/T-11 when you type them in, as well as the way numbers are displayed. If you enter a temporary radix control operator (Section 7.1.1), you override the current radix for the number you are typing but MDE/T-11 continues to display numbers in the current radix.

MDE/T-11 recognizes the following four radixes:

BINARY

OCTAL

DECIMAL

HEXADECIMAL

Consider the following command:

```
MDE> SET BREAK 1ØØØ<RET>
```

This sets the breakpoint at location 1ØØØ. Whether it is 1ØØØ binary, octal, decimal or hexadecimal depends on the setting of the radix mode. The default radix is OCTAL. If the radix is not set, the location is interpreted as 1ØØØ octal.

7.2.2 Display Mode

The following display modes specify the length and format of values.

BYTE - An 8-bit number

WORD - A 16-bit number

ASCII - Two ASCII characters in 16 bits (low-order byte followed by high-order byte)

INSTRUCTION - A PDP-11 instruction in as many words as required

RAD5Ø - Three Radix-5Ø encoded characters in 16 bits

Display modes do not affect the way input is interpreted. In the following example, suppose the current display mode is WORD and the current radix is OCTAL:

```
MDE>DEPOSIT 1000 = 136252<RET>
```

This command deposits the value 136252 in location 1000.

Now consider the following possibilities for different display modes.

Command	Result
EXAMINE/WORD 1000	1000 : 136252
EXAMINE/BYTE 1000	1000 : 252
EXAMINE/BYTE 1001	1001 : 274
EXAMINE/WORD 1001	ERROR (an odd address in WORD mode is illegal)

7.2.3 Address Mode

Whether or not MDE/T-11 displays addresses symbolically depends on the address mode setting.

```
SYMBOLIC

NOSYMBOLIC
```

If SYMBOLIC is in effect, addresses are displayed as symbols whenever possible; that is, in symbol+offset format, with the offset within a limit of 1024 bytes from the nearest symbol having a smaller value.

Conversely, if the current address mode setting is NOSYMBOLIC, addresses are displayed numerically in the current radix.

7.2.4 Mode Commands

The SET MODE command sets the modes. For example, the following command sets the radix mode to decimal:

```
MDE>SET MODE DECIMAL<RET>
```

If a mode is not set, the following defaults are in effect.

Mode	Default
Radix	OCTAL
Display	INSTRUCTION
Address	SYMBOLIC

The SHOW MODE command displays the mode settings. In response to this command, MDE/T-11 might display the following:

```
The current modes are:  OCTAL, SYMBOLIC, WORD
```

CANCEL MODE cancels any modes set and restores the initial default settings.

You can temporarily override a mode in some commands by typing a mode qualifier after the command. This override applies to output only, however. A mode qualifier consists of a value preceded by a slash (/). For example, if the display mode is WORD and you want to print the contents of a memory location in an instruction format, you can use the INSTRUCTION qualifier in the EXAMINE command.

```
MDE>EXAMINE /INSTRUCTION 1000:1002<RET>
```

With this command, the contents of locations 1000 through 1002 are displayed in the instruction format. The display mode, however, is not changed. The next command is interpreted according to the display mode, which in this case is WORD.

CHAPTER 8

COMMANDS

This chapter describes the format, functions, syntax, parameters and qualifiers of all MDE/T-11 commands. It also gives examples of these commands.

8.1 COMMAND FORMAT

The MDE/T-11 command format has four basic elements.

Command

Keyword

Qualifier

Parameter

Combined in a sequence, these elements tell MDE/T-11 what task to perform. In some commands, some of the elements may not be required.

The general format of an MDE/T-11 command is

```
command keyword /qualifier parameter !comment
```

where Command is a verb specifying the function to be performed.

Keyword is a noun or verb further specifying the function to be performed.

Qualifier is a word that modifies the command (some qualifiers take a numerical argument; an argument can be any single value or expression enclosed in parentheses).

Parameter is an expression on which the command and keyword operate.

Comment is an optional text message to be ignored by MDE/T-11.

In Section 8.2 the following conventions are used in presenting command syntax.

[ ] Brackets enclose one or more optional elements.

{ } Braces enclose two or more mandatory elements; you must choose one.

... Ellipses follow an element that can be repeated.

☐ A box encloses a default element or value or initial value.

EV Boldface characters denote abbreviations.

8.2 COMMANDS

This section describes the functions, syntax and basic elements of each MDE/T-11 command.

@file-spec

The @file-spec command invokes indirect commands from the file specified. The file must be a sequential ASCII file such as that produced by a text editor or the SET OUTPUT LOG and SET LOG commands. Indirect command files can be invoked wherever any other command can be given. You can use any valid MDE/T-11 command in an indirect command file.

You can nest indirect command files to three levels, unless you turn on the logging mechanism. If you turn on the logging mechanism, you can nest indirect command files to only two levels.

Syntax

@ file-spec invokes a command file.

Parameter

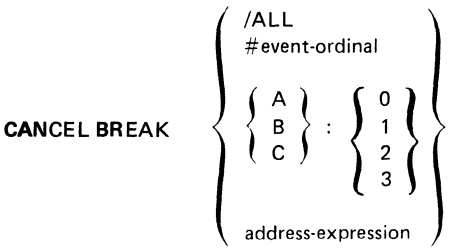
File-spec is the indirect command file to be executed. If you do not give a file extension, MDE/T-11 looks for a file with a .COM extension, and if unsuccessful tries the .CMD extension.

You can abort execution of an indirect command file at any time by typing two CTRL/Cs.

CANCEL BREAK

The CANCEL BREAK command cancels a breakpoint or all breakpoints. You must include one of the three available parameters (board:comparator, event-ordinal, address-expression) or the /ALL qualifier in this command.

Syntax



The shortest form of CANCEL BREAK is CAN BR

Parameters

Board:comparator specifies the state analyzer (A, B or C) and comparator (Ø, 1, 2 or 3) through which the breakpoint is set.

#event-ordinal is an identification number from Ø through 11 preceded by a number sign, assigned to the breakpoint by MDE/T-11.

Address-expression specifies the address of the breakpoint to be canceled.

Qualifier

/ALL cancels all breakpoints.

Examples

Command	Result
CAN BR #1	Cancels breakpoint specified by event ordinal #1
CAN BR START+2Ø	Cancels breakpoint at location symbolized by START+2Ø
CAN BR A:3	Cancels breakpoint assigned to comparator 3, analyzer A
CAN BR/ALL	Cancels all breakpoints

Related Commands

SET BREAK

SHOW BREAK

CANCEL MODE

The CANCEL MODE command cancels all modes and sets them to their default values (octal, symbolic, instruction).

Syntax

CANCEL MODE

The shortest form of CANCEL MODE IS CAN M

Example

Command	Result
CAN M	Sets all modes to default values (octal, symbolic, instruction)

Related Commands

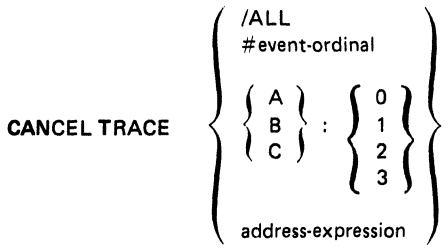
SET MODE

SHOW MODE

CANCEL TRACE

The CANCEL TRACE command cancels a tracepoint or all tracepoints. You must include one of the three available parameters (board:comparator, event-ordinal, address-expression) or the /ALL qualifier in this command.

Syntax



The shortest form of CANCEL TRACE is CAN T

Parameters

Board:comparator specifies the state analyzer (A, B or C) and comparator (Ø, 1, 2 or 3) through which the tracepoint is set.

#event-ordinal is an identification number from Ø through 11 preceded by a number sign, assigned to the tracepoint by MDE/T-11.

Address-expression specifies the address of the tracepoint to be canceled.

Qualifier

/ALL cancels all tracepoints.

Examples

Command	Result
CAN T #4	Cancels tracepoint associated with ordinal 4
CAN T START+6	Cancels tracepoint at location symbolized by START+6
CAN T/ALL	Cancels all tracepoints

Related Commands

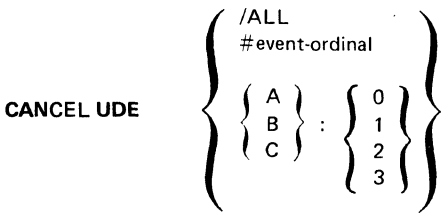
SET TRACE

SHOW TRACE

CANCEL UDE

The CANCEL UDE command cancels a user-defined event (UDE) or all such events. You must include one of the two available parameters (board:comparator, event-ordinal) or the /ALL qualifier in this command.

Syntax



The shortest form of CANCEL UDE is CAN UDE

Parameters

Board:comparator specifies the state analyzer (A, B or C) and comparator (Ø, 1, 2 or 3) used to set the event.

#event-ordinal is an identification number from Ø through 11 preceded by a number sign, assigned to the event by MDE/T-11.

Qualifier

/ALL cancels all user defined events.

Examples

Command	Result
CAN UDE A:2	Cancels event set in comparator 2, analyzer A
CAN UDE #5	Cancels event associated with ordinal 5
CAN UDE/ALL	Cancels all events

Related Commands

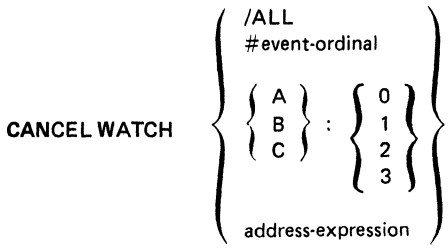
SET UDE

SHOW UDE

CANCEL WATCH

The CANCEL WATCH command cancels a watchpoint or all watchpoints. You must include one of the three available parameters (board:comparator, event-ordinal, address-expression) or the /ALL qualifier in this command.

Syntax



The shortest form of CANCEL WATCH is CAN W

Parameters

Board:comparator specifies the state analyzer (A, B or C) and comparator (Ø, 1, 2 or 3) through which the watchpoint is set.

#event-ordinal is an identification number from Ø through 11 preceded by a number sign, assigned to the watchpoint by MDE/T-11.

Address-expression specifies the address of the watchpoint to be canceled.

Qualifier

/ALL cancels all watchpoints.

Examples

Command	Result
CAN W #1	Cancels watchpoint associated with event ordinal 1
CAN WA SUM1	Cancels watchpoint at location symbolized by SUM1
CAN W/ALL	Cancels all watchpoints

Related Commands

SET WATCH

SHOW WATCH

CLEAR TRACERAM

The CLEAR TRACERAM command initializes the MICRO/T-11 bus cycle trace buffer.

Syntax

CLEAR TRACERAM

The shortest form of CLEAR TRACERAM is CLE TRACER

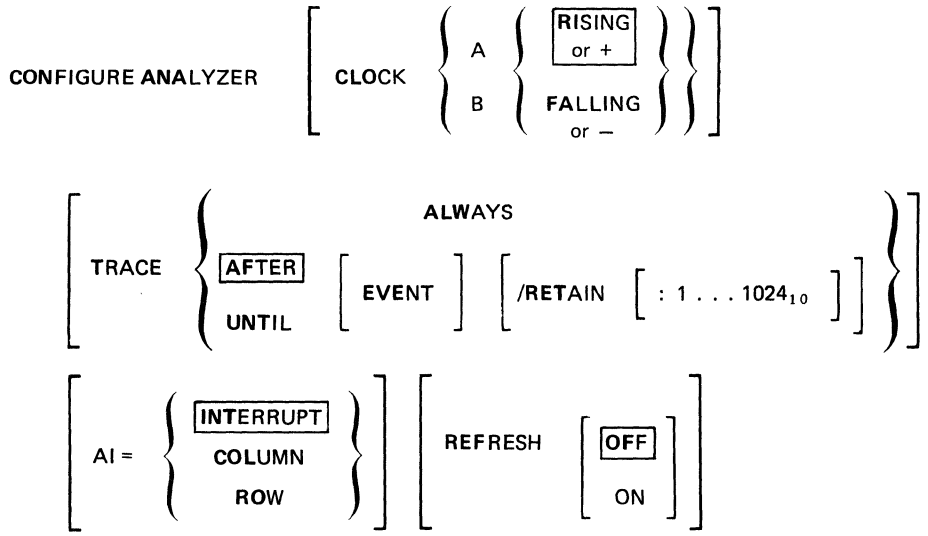
Example

Command	Result
CLE TRACER	Clears trace buffer; initializes RETAIN count if any was specified in CONFIGURE ANALYZER command

CONFIGURE ANALYZER

The CONFIGURE ANALYZER command determines the modes of operation of the state analyzer. These modes include the clock edge at which external probe lines are sampled, the trace-RAM operation, and the time at which AI lines are sampled by the state analyzer.

Syntax



The shortest form of CONFIGURE ANALYZER is CON ANA

Parameters

CLOCK causes sampling of external probe information to occur at the RISING (+) or FALLING (-) edge of the externally supplied probe clock signal. The RISING parameter causes the probe data to be latched on the rising edge of the probe clock signal. The FALLING parameter causes the probe data to be latched on the falling edge of the probe clock signal.

TRACE determines how bus cycles are captured in the trace RAM. If you specify

TRACE ALWAYS — Every bus cycle is captured and TRACE and STOP event actions are ignored.

TRACE AFTER — Tracing begins when an event with a TRACE action occurs.

TRACE UNTIL — Tracing begins with program execution.

With TRACE AFTER or TRACE UNTIL, tracing continues until an event with a STOP action occurs or program execution ends or is halted. EVENT is an optional keyword that does not affect the command. The default trace-RAM mode is TRACE AFTER EVENT with no RETAIN limit.

AI determines when the AI lines are sampled; the default is INTERRUPT.

ROW	Leading edge of RAS
COLUMN	Leading edge of CAS
INTERRUPT	Trailing edge of CAS

REFRESH ON captures memory refresh cycles in the trace RAM. The default is OFF. This parameter applies only when the MICRO/T-11 is in a dynamic mode.

Qualifier

/RETAIN causes the trace RAM to stop after the number of cycles specified by its argument have been collected. With no argument, the retain count is 1024 (decimal). If /RETAIN is not given, tracing continues until stopped for some other reason (described previously).

Examples

Command	Result
CON ANA AI=ROW	AI lines examined when ROW information is present
CON ANA TRACE ALWAYS AI=ROW	AI lines examined when ROW information is present; all bus cycles captured
CON ANA TRACE UNTIL EVENT /RETAIN:20	Tracing begins on execution; ends after 20 bus cycles are captured, when an event with STOP action occurs or when emulation is halted.

Related Commands

CLEAR TRACERAM

DISPLAY TRACERAM

SET UDE

CONFIGURE CLOCK

The CONFIGURE CLOCK command selects the processor clock signal from the emulator's 5.Ø688 MHz crystal-controlled source or from an external source connected by the EXTERNAL CLOCK connector. If you want either of these options, you must place the pod switches in the ON position. The MICRO/T-11 must be powered down to use this command.

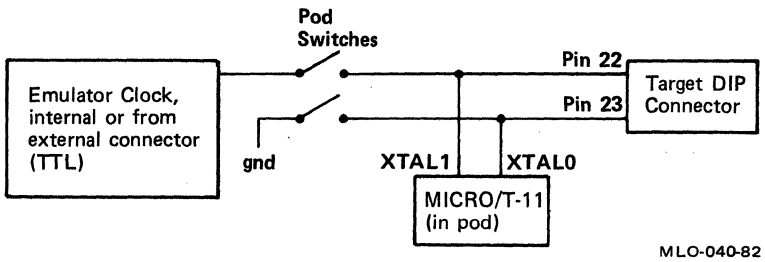


Figure 8-1 Software-Selected Emulator Clock

As shown in Figure 8-1, the target clock signal supplied from the emulator (switches on) appears on the XTAL lines of the MICRO/T-11. To avoid interfering with the target clock signal, observe the following precautions:

1. Do not drive the XTAL lines or connect them to circuitry that could interfere with the clock signal.

2. Use caution when using the clock signal to drive TTL inputs so as not to overload or interfere with the signals.

You can also select the clock signal produced on the target hardware by placing the two switches on the MDE/T-11 pod in the OFF position (Chapter 6). When this is done, the CONFIGURE CLOCK command does not control the processor clock source.

Syntax

CONFIGURE CLOCK { **EMULATOR** | **EXTERNAL** }

The shortest form of CONFIGURE CLOCK is CON CLO

Parameters

EMULATOR (the default) selects the emulator's 5.Ø688 MHz clock.

EXTERNAL selects the input signal on the EXTERNAL CLOCK connector.

When you select the EXTERNAL parameter, the signal input into the EXTERNAL clock connector must meet MICRO/T-11 clock specifications for TTL clock input; the signal must be a TTL-level, low-noise, 5Ø% duty-cycle square wave.

Examples

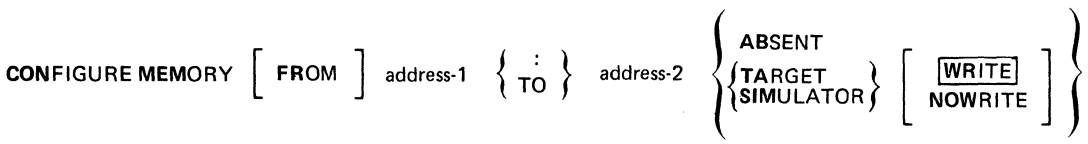
Command	Result
CON CLO EMULATOR	Selects 5.Ø688 MHz processor clock
CON CLO EXTERNAL	Selects externally generated processor clock

CONFIGURE MEMORY

The CONFIGURE MEMORY command defines MICRO/T-11 address space in the target or in the memory simulator. Memory is mapped and protected in 256-byte increments, and each increment can be write-protected. Addresses not configured with this command are considered absent from MICRO/T-11 address space. Addresses configured with this command are aligned along 256-byte boundaries.

See Section 3.3 for an explanation of memory allocation and for a description of restrictions in the use of simulator memory when one memory simulator is used.

Syntax



The shortest form of CONFIGURE MEMORY is CON MEM

Parameters

Address-1 and address-2 specify the low and high limits of the memory range to be mapped and protected.

TARGET configures memory in the target, SIMULATOR configures memory in the simulator, and ABSENT specifies that memory cannot be read from or written to.

WRITE specifies that memory may be written to, and NOWRITE write-protects memory (write protection does not change contents of protected location). The default is WRITE.

Initially, all memory is configured ABSENT.

Examples

Command	Result
CON MEM FR 1000 TO 1777 SIMULATOR NOWRITE	Reserves two segments in simulator with NOWRITE access for locations 1000 thru 1777
CON MEM FR 2000 TO 4777 TARGET	Reserves six segments in target with WRITE access for locations 2000 thru 4777
CON MEM FR 0 TO 5777 ABSENT	Designates locations 0 thru 5777 ABSENT; prohibits access to these locations for reading and writing

Related Commands

COPY

LOAD

SHOW CONFIGURE

CONFIGURE MODE

The CONFIGURE MODE command selects operating and timing features of the MICRO/T-11 by setting bits in the mode register. This command also determines the source of the mode register. The MICRO/T-11 must be powered down when you use this command.

You can direct MDE/T-11 to configure mode bits in the mode register in the MICRO/T-11 emulator or to declare the setting of the mode bits in the target hardware. The MICRO/T-11 emulator will execute using the mode register source specified. This command lets you emulate the MICRO/T-11 in all its operating modes.

Number of bits -- You can select the 8-bit or 16-bit data bus configuration.

Memory refresh -- You can select static (no refresh) or dynamic (refresh). If you select dynamic, you can specify large (64K) or small (4K or 16K) memory chips depending on your application hardware.

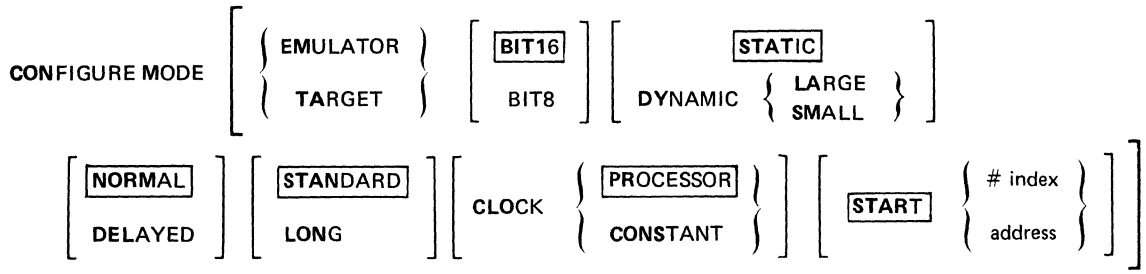
Triggering -- You can specify normal or delayed read/write cycles, and standard or long pulses for bus cycle timing.

Clock -- You can select a normal processor clock frequency (supplied as directed in your CONFIGURE CLOCK command) or a constant clock (one-half the normal processor clock frequency).

Start/Restart address

See the MICRO/T-11 User's Guide for a complete description of MICRO/T-11 operating modes.

Syntax



The shortest form of CONFIGURE MODE is CON M

Parameters

TARGET takes the mode register from the target, and EMULATOR takes the mode register from the emulator. You must specify one or the other. If you specify TARGET, you must also specify any of the remaining parameters that are not defaults.

BIT8 selects a bus width of 8 bits and BIT16 a bus width of 16 bits. The default is BIT16.

STATIC, DYNAMIC SMALL or DYNAMIC LARGE selects the memory refresh mode. The default is STATIC.

NORMAL specifies standard timing-pulse triggering. DELAYED uses the read or write assertion needed to make the MICRO/T-11 bus a proper subset of the 8080 bus. The default is NORMAL.

STANDARD specifies the standard timing-pulse width. LONG adds 133 ns to the standard pulse width for slower-access peripheral circuits. The default is STANDARD.

CLOCK gives the clock assertion rate. PROCESSOR asserts the clock once for every microinstruction. CONSTANT asserts the clock at a constant rate (half the input frequency). The default is CLOCK PROCESSOR.

START determines the power-up address to be used. The default is START 140000. You can specify the start address by its index (first column) preceded by a number sign (#), or by the start address directly (second column).

Index	Start Address	Restart Address
0	140000	140004
1	100000	100004
2	40000	40004
3	20000	20004
4	10000	10004
5	0	4
6	173000	173004
7	172000	172004

If the MICRO/T-11 is powered up and you did not issue a CONFIGURE MODE command, the MICRO/T-11 mode register bits are read from the emulator and the defaults are BIT16, STATIC, NORMAL, STANDARD, CLOCK PROCESSOR and START 140000 (#0).

Examples

Command	Result
CONF MODE EMULATOR BIT8 DYNAMIC SMALL	Emulates mode register using default settings except for 8-bit bus width and dynamic memory refresh (small mode)
CONF MODE TARGET DELAYED LONG	Reads mode register from target; declares its contents to be default settings except for delayed-long pulse options

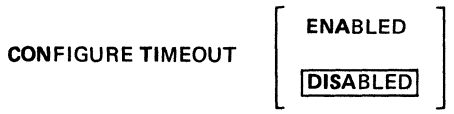
Related Command

SHOW CONFIGURE

CONFIGURE TIMEOUT

The CONFIGURE TIMEOUT command enables or disables the emulator processor-timeout detection feature. When enabled, this feature lets the emulator detect the processor's failure to assert a fetch within a fixed time (approximately Ø.6 seconds). The emulator is initialized with this feature disabled.

Syntax



The shortest forms of CONFIGURE TIMEOUT are CON TI ENA and CON TI DISA

Parameters

ENABLED turns on the processor timeout feature.

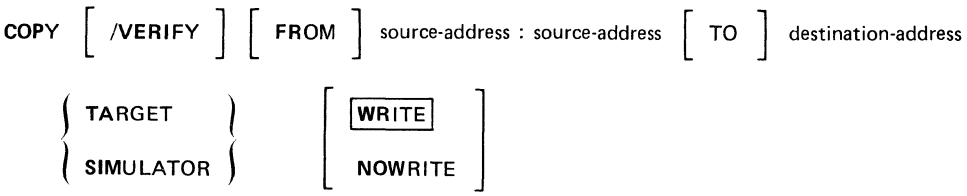
DISABLED (the default) turns off the timeout feature.

COPY

The COPY command copies data or code from one place in the target address space to another. You will find this command especially useful in editing the contents of the target ROM. To edit the target ROM, use the COPY command with the SIMULATOR and NOWRITE parameters. This command line copies the contents of the target ROM into the memory simulator at the same addresses as those in the target ROM.

This command performs an implicit CONFIGURE MEMORY command for the destination memory. If destination memory is already configured, any data there is lost, and 256-byte blocks containing source-start-address through source-end-address are copied to the destination address.

Syntax



The shortest form of COPY is COP

Qualifier

/VERIFY performs a write-check during the copy operation. If you do not specify /VERIFY, no write-check is performed.

Parameters

Source-start-address, source-end-address and destination-start-address are expressions. Source-start-address specifies the first address of the source, source-end-address specifies the last address of the source, and destination-start-address specifies the first destination address. All three addresses are rounded to 256-byte boundaries as in the CONFIGURE MEMORY command.

TARGET or SIMULATOR specifies the memory to be used for the destination.

WRITE or NOWRITE specifies the protection to be asserted for the destination memory. The default is WRITE.

Source-start-address and source-end-address define a range. This range must be configured prior to using the COPY command. The following restriction applies if the source and destination are located in the same memory.

source-end < destination-start

or

destination-start < source-start

Examples

Command	Result
COP /VERIFY 1000:1200 4000 TARGET	Configures 4000 thru 4377 in target memory as WRITE; copies from 1000 to 4000, 1001 to 4001, ... 1377 to 4377; verifies data in destination
COP 7610:7612 14000 SIM NOWRITE	Configures 14000 thru 14376 in simulator memory as NOWRITE; copies 256 bytes with write-check as 7400 to 14000, 7401 to 14001, ... 7777 to 14377
CONFIGURE MEMORY 100000:137777 TARGET **COPY 100000:137777 TO 100000 SIM NOWRITE**	Assumes configured ROM memory on target; simulates and write-protects ROM space; copies contents of target ROM into simulator memory at same addresses; allows editing of ROM contents

DEFINE

The DEFINE command defines a new symbol as a given value, or redefines a symbol loaded from the program symbol table.

Syntax

DEFINE symbol = address-expression

The shortest form of DEFINE is DEF

Parameter

Address-expression is a value to be associated with a symbol. MDE/T-11 evaluates address-expression when the command is executed, and associates the value of the expression with the symbol. From that point on, you can use the symbol in place of the value.

Examples

Command	Result
DEF V = 20000	Defines V as address 20000
DEF V3 = V + 6	Defines V3 as address symbolized by V + 6 (20006)
DEF V3 = V3 + 1	Increments the defined value of V3

Related Commands

SHOW DEFINE

UNDEFINE

DEPOSIT

The DEPOSIT command changes the contents of a specified memory location or register, or a range of memory locations or registers. This command can fill a range of locations with a single value, or distribute a sequence of values over a range of locations. You can also use the DEPOSIT command to assemble PDP-11 instructions and to deposit the assembled instructions into memory.

Syntax

DEPOSIT [/VERIFY] [/BYTE /WORD] [destination [: destination]] = value [,value, . . .]

The shortest form of DEPOSIT is D

Parameters

Destination is an address-expression, a register, or a range of address-expressions or registers. If destination is omitted, data is deposited into the memory locations immediately following the last locations specified by the most recent EXAMINE or DEPOSIT command. This means you can deposit sequences of PDP-11 instructions or data in contiguous locations without having to type an address each time. If a deposited instruction takes up more than one word, MDE/T-11 keeps track of the location for the next deposit.

Value is an expression, a PDP-11 instruction enclosed in quotes or an ASCII string enclosed in apostrophes.

Qualifiers

/BYTE or /WORD specifies the size of the storage locations used in the deposits. The default is WORD unless the current display mode is BYTE.

/VERIFY performs a write-check on each value deposited in memory.

MDE/T-11 evaluates the first expression following the equal sign, and stores the value of that expression at the address specified by destination. If a second expression is given, that expression is evaluated and its value is stored in the word or byte address following the one specified (depending on the setting of the length qualifier).

This process continues for any other expressions in the list. Each time the contents of an address are changed, the current address is incremented to receive the next value. If you specify a range of addresses with the number of values less than the number of addresses in the range, the sequence of values specified for deposit begins repeating. If you specify a range of registers, your value list must contain the same number of values as the addresses specified. For registers, the VT100 static display is updated and the contents of the updated registers are highlighted.

You can include PDP-11 instructions in source language form in your list of values, but you must enclose them in quotation marks. These instructions generate code as if they were assembled in an absolute program section. For example

```
DEP 1000=1,2,"MOV @#START,3(R0)",4
```

This command deposits the values 1, 2, 013760, START, 3 and 4 into locations 1000 through 1012 (octal).

When using PDP-11 instructions, you can include

- Multiple PDP-11 instructions in a single command
- Labels (an implicit DEFINE is performed)
- PDP-11 instruction deposits to an even-numbered address when the /BYTE qualifier is used or the byte mode is in effect
- Comment text preceded by a semicolon (ignored by assembler)

However, you cannot include

- PDP-11 instruction deposits to ranges (such as DEP 0:100=), to registers, or to add memory addresses
- MACRO-11 relocations and other operations of the MACRO-11 assembler involved in generating relocatable code
- MACRO-11 assembler directives and other MACRO-11 elements
- Blanks within MACRO-11 expressions and operands
- MACRO-11 angle brackets (< >) in expressions, or arithmetic operators other than + or -
- A radix override (for example, %O'123')

PDP-11 instruction syntax is the same as for MACRO-11 assemblers. However, when using expressions to represent index offsets or the immediate mode, the expressions are limited to a series of terms, each preceded or followed by the + or - arithmetic operator. In addition, terms used in these expressions can consist of

- Numbers in the current MDE/T-11 radix mode (numbers immediately followed by a period are interpreted as decimal)
- The MACRO-11 current location operator (.), not the MDE/T-11 current address indicator
- Any valid symbol
- The value of any single ASCII character, preceded by an apostrophe (for example, MOV #'Q,R0)

If your DEPOSIT command changes the value of an address specified as a watchpoint, the watchpoint is not triggered.

Examples

Command	Result
DEPOSIT SUM1=2Ø	Sets contents of location SUM1 to 2Ø
DEPOSIT V:V+12=Ø	Fills memory locations V, V+2, ... V+12 with Ø
DEPOSIT V+2=1Ø,11	Sets contents of location V+2 to 1Ø, and contents of location V+4 to 11
D . = 14	Sets contents of location last used in a DEPOSIT or EXAMINE command to 14
DEPOSIT 1ØØ = 1,2,3,4	Sets contents of memory location 1ØØ to 1, 1Ø2 to 2, 1Ø4 to 3 and 1Ø6 to 4, assuming the current mode is not BYTE
DEPOSIT 2ØØ:22Ø = 1,Ø	Sets contents of memory locations 2ØØ, 2Ø4, 21Ø, ... to 1, and memory locations 2Ø2, 2Ø6, 212 ... to Ø, assuming the current display mode is not BYTE
DEPOSIT RØ:R2 = 1,2Ø,4	Sets contents of RØ to 1, R1 to 2Ø and R2 to 4
DEPOSIT 1ØØØ = "MOV @#START,3(RØ)"	Sets contents of word locations 1ØØØ, 1ØØ2 and 1ØØ4 to assembled value of PDP-11 instruction MOV @#START,3(RØ)
DEPOSIT 1ØØ = 'STRING'	Deposits the ASCII characters "STRING" into the 6 bytes starting at address 1ØØ.

Related Command

EXAMINE

DISPLAY TRACERAM

The DISPLAY TRACERAM command displays the contents of the trace RAM. Two CTRL/C characters abort execution of this command and return MDE/T-11 to the command level. A short delay may occur after the CTRL/Cs are typed.

Any part or all of the trace RAM can be displayed when the MICRO/T-11 is in the pause state, but access to the trace RAM is limited when the MICRO/T-11 is running. You are limited to 16 frames of bus cycle data when using a hard-copy terminal; 13 frames when using a VT100. If the MICRO/T-11 is running and event detection is disabled for a short time, some events can be lost.

Syntax

DISPLAY [/radix-mode] [/address-mode] **TRACERAM** [start] [window]

The shortest form of DISPLAY TRACERAM is DI TRACER

Parameters

Start is a number in the range 0 through 1023 (decimal) specifying the starting location within the trace RAM. Start is interpreted as a decimal number; it must be greater than or equal to window - 1.

Window specifies the number of bus cycles to display, counting from start. The default is 16 for hard-copy terminals or 13 for VT100 terminals. Window is interpreted as a decimal number; it must be less than or equal to 1024.

Qualifiers

/radix-mode is the desired radix mode. The current radix is the default.

/address-mode specifies the display for the address field (SYMBOLIC, NOSYMBOLIC).

Examples

Command	Result
DI TRACER 12 10	Displays 10 cycles, starting at the 12th cycle before the last
DI TRACER	Displays the last 17 (decimal) cycles in the current radix and type
DI TRACER/DEC	Displays the last 17 cycles in decimal

MDE/T-11 prints a header line and then the cycle number in decimal and the fields of the bus cycle in the current radix. For example

Frame No.	Bus Trans Status	Address	Data	SEL Lines	AI Lines	Probe Lines A
16	Fetch	START	12706	01	11111111	00000000
15	Read	START+2	40000	00	11111111	00000000
14	Fetch	START+4	5067	01	11111111	00000000
13	Read	START+6	177764	00	11111111	00000000
12	Read	SUM1	26	00	11111111	00000000
11	Write	SUM1	0	00	11111111	00000000
10	Fetch	START+10	5067	01	11111111	00000000
9	Read	START+12	177762	00	11111111	00000000
8	Read	SUM2	12	00	11111111	00000000
7	Write	SUM2	0	00	11111111	00000000
6	Fetch	START+14	12701	01	11111111	00000000
5	Read	START+16	10	00	11111111	00000000
4	Fetch	START+20	12700	01	11111111	00000000
3	Read	START+22	40000	00	11111111	00000000
2	Fetch	START+24	61067	01	11111111	00000000
1	Read	V	5	00	11111111	00000000
0	Read	START+26	177744	00	11111111	00000000

Bus Trans Status is a literal field indicating the MICRO/T-11 cycle type, which is one of

FETCH
READ
WRITE
IACK
ASPI
READ-DMA
WRITE-DMA
UNKNOWN

Address is printed symbolically if the current address mode is SYMBOLIC. Symbol names are truncated on the right to fit in a 17-character field allowing for possible offsets.

Data is a 16-bit field displayed in the current radix. Data will always be less than 377 (octal) when the MICRO/T-11 is operating in the 8-bit mode.

SEL Lines is a 2-bit binary field.

AI Lines is an 8-bit binary field.

Probe Lines A is an 8-bit binary field. If your system contains two state analyzers, Probe Lines B is also displayed.

EVALUATE

The EVALUATE command performs a computation. You can use this command to check expressions before using them in other commands.

Syntax

EVALUATE [/radix-mode] expression

The shortest form of EVALUATE is EV

Qualifier

Radix-mode temporarily overrides the current radix mode when displaying the result of the evaluation, but it does not affect the interpretation of numbers in the expression to be evaluated. The radix mode can be OCTAL, DECIMAL, HEXADECIMAL or BINARY.

Examples

Command	Result
EV .+4	Displays memory address designated by .+4; period (.) specifies last location used in DEPOSIT or EXAMINE command
EVAL /HEX START+2	Displays memory address designated by START+2 in hexadecimal

MDE/T-11 responds by printing the value of each expression after an equal sign. For example

```
MDE> EVAL START+2<RET>
= 40026
MDE> EVAL/HEX START+2<RET>
= 4016
MDE>
```

EXAMINE

The EXAMINE command displays the contents of a specified memory location or the contents of a range of locations, and the contents of the MICRO/T-11 registers.

Syntax

EXAMINE [/mode . . .] [argument [,argument . . .]]

The shortest form of EXAMINE is E

Parameter

Argument can be an expression, a register, or a range of expressions or registers. A range of expressions has the format

expression-1:expression-2

The address for expression-1 must be less than or equal to the address for expression-2. This displays the contents of locations starting with expression-1 and continuing through expression-2.

If argument is omitted, MDE/T-11 examines the locations following the last location specified in the most recent EXAMINE or DEPOSIT command.

Qualifier

/mode overrides the current radix, display or address mode. You can use

Radix: OCTAL, DECIMAL, HEXADECIMAL or BINARY

Display: BYTE, WORD, INSTRUCTION, ASCII, or RAD5Ø

Address: SYMBOLIC or NOSYMBOLIC

Examples

```
MDE>EXAMINE START:START+4<RET>

                START : BR  START+2
              START+2 : BEQ TTYSET
              START+4 : MOV #4ØØØØ,SP

MDE>EXAMINE/WORD/DECIMAL 1ØØ:11Ø<RET>

MDE>EXAMINE/ASCII   STRING:STRING+1ØØ<RET>
```

EXIT

The EXIT command ends a degugging session, returning to the operating system command level. This command also closes any log files and indirect command files. EXIT does not affect the state of the MICRO/T-11.

Syntax

```
EXIT
```

The shortest form of EXIT is EXI

Example

Command	Result
EXIT	Leaves MDE/T-11 and returns to operating system

GO

The GO command takes the MICRO/T-11 out of the pause state and lets it execute code. If no address is supplied, the program starts at the current PC. The MICRO/T-11 must be powered up before program execution can be started. See the CONFIGURE MODE and POWER commands for additional information.

Syntax

GO [address-expression]

The shortest form of GO is G

Parameter

Address-expression is the parameter for this command. Address-expression specifies the address at which execution of the program begins.

Examples

Command	Result
GO START	Begins execution of program at START
GO	Begins execution of program at address given in PC

Related Commands

HALT

SHOW TARGET

STEP

HALT

The HALT command stops execution of your program in the target processsor and displays that processor's registers.

Syntax

```
HALT
```

The shortest form of HALT is HA

Examples

Command	Result
HALT	Stops execution of program and displays registers

MDE/T-11 responds by giving the address after which execution was halted (PC), the contents of registers (RØ through R5), the stack pointer (SP) and the processor status word (PS). Values are given in the current radix. If the current address mode is SYMBOLIC, the PC is displayed symbolically. For example

```
MDE>HALT
;MDE-I-PROEXEHLT, Processor execution halted at PC:START+24

RØ:       40000
R1:          10
R2:           3
R3:      177777
R4:      177777
R5:      177777
SP:       40000
PC:    START+24
PS:         340
```

If you are not using a VT100, registers and their contents are listed serially as shown above. If you are using a VT100, register contents are displayed in the static region of the screen.

Related Commands

GO

SHOW TARGET

HELP

The HELP command displays information about MDE/T-11 commands including command notation and suggestions for using system features.

Syntax

HELP [topic [subtopic . . .]]

The shortest form of HELP is HE

Examples

Command	Result
HELP	Displays message listing command verbs and nouns
HELP EXAMINE	Displays message about EXAMINE command
HELP SET	Displays message giving various forms of SET command
HELP SET BREAK	Displays message about SET BREAK command
HELP SET *	Displays messages about all SET commands
HELP S	Displays messages about all commands beginning with S

INITIALIZE

The INITIALIZE command initializes the trace RAM and resets MDE/T-11 hardware to its default modes of operation. These modes result in

All memory configured as absent

All set events canceled

The trace-RAM mode set to AFTER with no RETAIN count

MICRO/T-11 powered down

Fetch timeout disabled

Processor clock taken from emulator (5 MHz)

Whenever you invoke MDE/T-11, an implicit CONFIGURE MODE EMULATOR (with all default settings) is performed.

Syntax

INITIALIZE

The shortest form of INITIALIZE is INI

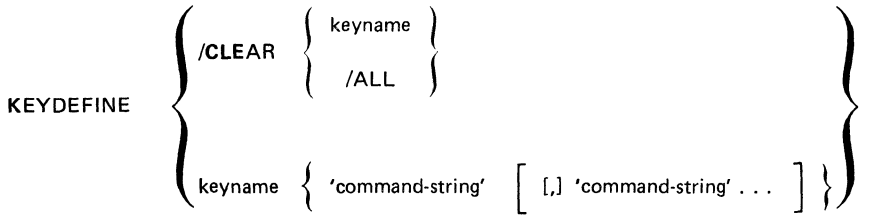
Example

Command	Result
INI	Initializes trace RAM and resets MDE/T-11 hardware to default modes of operation

KEYDEFINE

The KEYDEFINE command defines a keypad PF key by associating one or more commands with it, or undefines one or all PF keys. This command can be used only in the VT1ØØ mode.

Syntax



The shortest form of KEYDEFINE is K

Parameters

Keyname is PF1, PF2, PF3 or PF4.

Command-string is any valid MDE/T-11 command. Up to eight quoted strings can be given.

If KEYDEFINE specifies a keyname and a sequence of one or more quoted command strings, the key is associated with the command sequence. After execution of this command, pressing the designated key causes the specified sequence of commands to be executed. These commands are echoed at the terminal and copied to the log file if logging is enabled.

Qualifiers

/CLEAR used alone undefines one PF key. /CLEAR and /ALL used together undefine all four PF keys. After execution of /CLEAR, pressing the designated key or keys produces an error.

Examples

Command	Result
KEY PF1 'E RØ'	Associates command 'E RØ' with PF1 key
KEY PF2 'E 1ØØ1:1Ø2Ø' 'GO RESTART'	Associates EXAMINE and GO commands with PF2 key
KEY /CLEAR PF1	Undefines PF1 key
KEY/CLEAR /ALL	Undefines keys PF1 thru PF4

Related Command

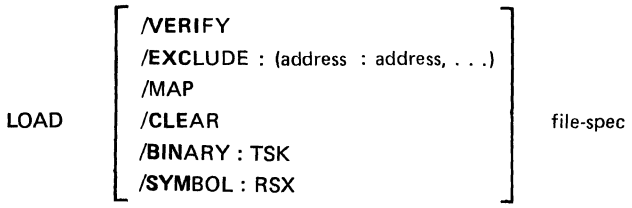
SET TERMINAL

LOAD

The LOAD command transfers data and code from disk files into MICRO/T-11 address space, and loads global program symbols from disk files into the MDE/T-11 symbol table. You can also use LOAD to deposit your program's transfer address into the MICRO/T-11 program counter (PC).

LOAD recognizes the RSX-11M Task Builder (TKB) memory image file produced by TKB (this file can have a .TKB or .EXE extension). LOAD also recognizes the symbol table file (.STB). You must configure memory prior to loading the .TSK (.EXE) or .STB file.

Syntax



The shortest form of LOAD is L

Qualifiers

/VERIFY performs a read-back check on each byte loaded into MICRO/T-11 address space.

/BINARY:type specifies that the file named will be loaded as a TKB memory image file (:TSK) or RT-11 memory image file (:LDA), as specified, regardless of the file specification.

NOTE

Under VAX/VMS, TKB (by default) produces files with an .EXE extension. If MDE/T-11 fails to find a .TSK file and no extension is specified, MDE/T-11 looks for a file with an .EXE extension.

/SYMBOL:type specifies that the file named be loaded as a symbol table file. "Type" indicates whether the file was generated by the RT-11 linker (:RT) or by the RSX-11M linker (:RSX).

/EXCLUDE:(address:address,...) prevents MDE/T-11 from loading any addresses specified in the exclude list. An exclude list is a list of address ranges. You enclose all items in the exclude list in parentheses and separate them with commas. Ranges you declare in the exclude list are interpreted by MDE/T-11 as ranges of bytes.

You will find this qualifier useful if you have configured a large portion or all of MICRO/T-11 address space, but require only a small portion of the address space to load your program.

Using the /EXCLUDE qualifier, you can load .TSK (.EXE) files quickly, because you exclude all irrelevant data from the .TSK (.EXE) file.

/MAP loads only those sections of an image file for which memory is configured. If memory is not configured, nothing is loaded. However, if the MICRO/T-11 is powered up and the image contains a transfer address, the transfer address is loaded into the MICRO/T-11 PC.

/CLEAR loads symbols from a program symbol table. This qualifier causes MDE/T-11's symbol table to be cleared (program and DEFINE symbols) before your program symbols are loaded. When you use this qualifier, you speed up the loading process considerably, because MDE/T-11 does not have to check for duplicate entries when loading your program symbols into its symbol table.

Parameter

File-spec can be any of the following.

An unquoted file name with no extension -- A default extension is supplied by MDE/T-11 if a /BINARY or /SYMBOL qualifier is used. If no such qualifier is used, the .TSK (.EXE) file and the .STB file are loaded as memory image and symbol table respectively.

An unquoted file name with a .TSK (.EXE) or .STB extension -- This file is loaded as a binary or symbol file as implied by the extension.

NOTE

Specifying a .STB extension without a /SYMBOL:RT qualifier causes MDE/T-11 to assume that the .STB file was produced by TKB and not LINK.

An unquoted file name with a nonstandard extension -- This file is loaded as specified in the required /BINARY or /SYMBOL qualifier.

A full file specification set in single quotes -- You must supply a file type qualifier (/BINARY or /SYMBOL). Single quoted file-specs are passed directly to the VAX/VMS operating system.

Examples

Command	Result
LOAD PROG	Loads file PROG.TSK (.EXE if .TSK file not found) into MICRO/T-11 address space (entire task image is loaded); loads symbols from symbol table file PROG.STB into MDE/T-11 symbol table
LOAD PROG.TSK	Loads memory image file into MICRO/T-11 address space
LOAD PROG.STB	Loads symbol table file into MDE/T-11 internal symbol table; PROG.STB is assumed to be in TKB format

LOAD/BINARY:EXE PROG

Loads file PROG.EXE (.EXE if .TSK file not found) as memory image file into MICRO/T-11 address space

LOAD/BINARY:EXE PROG.XYZ

or

LOAD/BINARY:EXE 'PROG.XYZ'

Loads file PROG.XYZ as memory image file into MICRO/T-11 address space

LOAD/SYMBOL:RT TEST.STB

Loads symbols from LINK-generated .STB file (TEST.STB)

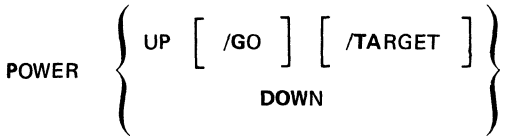
LOAD PROG.XYZ Illegal (MDE/T-11 does not attempt to parse quoted file specs, thus cannot determine file type)

LOAD 'PROG.XYZ' Illegal (MDE/T-11 cannot determine file type)

POWER

The POWER command lets you simulate a power-on condition and direct the MICRO/T-11 to begin execution or enter the pause state. This command also lets you select the power-on initialization (PUP) signal from the MDE/T-11 system (for simulated power-on) or the target system.

Syntax



The shortest form of POWER is P

Parameters

POWER UP asserts the PUP signal (briefly sets high, then low), causing the MICRO/T-11 to read the mode register.

POWER DOWN deasserts the PUP signal (sets high), causing the MICRO/T-11 to become inactive.

Qualifiers

/GO tells the system not to enter the pause state on power-up and to begin execution immediately (at the start address).

/TARGET takes the power-up signal from the target. The default takes the power-up signal from the emulator.

Examples

Command	Result
P UP	PUP line set high, then low; MICRO/T-11 pauses at start address specified by CONFIGURE MODE command
P DOWN	PUP line set high
P UP/GO	PUP line set on; execution of program starts at start address
P UP/GO/TAR	Source of PUP line switches to target hardware

Related Commands

CONFIGURE

SHOW TARGET

RESET ANALYZER

The RESET ANALYZER command initializes all comparator event counters and software event counters to the /COUNT and /AFTER values specified previously, and clears all event and qualifier flags.

Syntax

RESET [ANALYZER]

Resets all event counters, and event and qualifier flags.

The shortest form of RESET ANALYZER is RESE

Example

Command	Result
RESET	Initializes all comparator and software event counters; clears all event and qualifier flags

Related Commands

SET BREAK

SET TRACE

SET UDE

SET WATCH

SET BREAK

The SET BREAK command sets a breakpoint, which is a point in the program where MICRO/T-11 target processor execution is halted.

Syntax

SET BREAK [**/COUNT:** [1] .. 256_{10} | **/AFTER:** [1] .. 65535_{10}] address-expression

The shortest form of SET BREAK is SE BR

Parameter

Address-expression is the address of an instruction. The breakpoint occurs after the instruction at that address is executed.

If a breakpoint exists at the specified address, the breakpoint is reestablished with the attributes given or implied by the new command. If a tracepoint exists at that address, the tracepoint is canceled and the breakpoint is set.

Qualifier

/COUNT:n signals the breakpoint every nth time it occurs, where n is any number in the range of 1 to 256. The default assumes a count of 1, and the breakpoint is signaled each time it occurs.

If /AFTER:n is given, signaling of the breakpoint causes the n to be decremented (n is any number in the range of 1 to 65535). If decremented to Ø, the breakpoint is reported; if not, an implicit GO is performed.

/COUNT is hardware-implemented and faster, but limited to 256 (decimal). /AFTER is software-implemented and much slower, but can be as large as 65535 (decimal). In addition, the /AFTER counter is displayed with the SHOW BREAK command. You can use the /COUNT and /AFTER counts together in this command. If you use both of these counts in the SET BREAK command, the event count can be extended to a maximum of 16,776,96Ø.

Examples

Command	Result
SE BR START	Breakpoint set at instruction labeled START
SE BR /COU:5 START	Breakpoint set at instruction labeled START to be signaled every 5th time instruction is executed

MDE/T-11 responds to the occurrence of a breakpoint by printing BREAK at PC = expression. For example

```
MDE>
BREAK at PC = START+10
```

Related Commands

CANCEL BREAK

RESET ANALYZER

SHOW BREAK

SET LOG

The SET LOG command specifies the name of a log file used to record a sequence of commands and responses. The logging mechanism is turned on and off by SET OUTPUT commands. A log file can be used as an indirect command file to recreate a debugging session. This command can be given as many times as desired in a debugging session to direct output to different log files.

Syntax

SET LOG file-spec

The shortest form of SET LOG is SE LOG

Parameter

File-spec may be up to 50 characters long, and it must consist of valid Radix-50 characters. The default uses CDS.LOG for the log file.

If you give file-spec without quotes and with no extension, MDE/T-11 uses a default extension of .LOG. If you give it with quotes, MDE/T-11 passes the enclosed string to the operating system but does not make any validity checks on the string.

Examples

Command	Result
SE LOG TEST.DTA	Designates TEST.DTA as log file
SE LOG TEST	Designates TEST.LOG as log file
SET LOG 'DK1:TEST.LOG'	Designates DK1:TEST.LOG as log file

The log file contains MDE/T-11 commands and responses. Responses are preceded by the comment indicator (!) so the log file can be used as an indirect command file. For example

```
MDE>SHOW BREAK<RET>
! #0 A:0 Breakpoint ...
.
.
.
```

Related Commands

SET OUTPUT

SHOW OUTPUT

SET MODE

The SET MODE command sets the radix, display and address modes.

Syntax

SET MODE { mode } [,mode , . . .]

The shortest form of SET MODE is SE M

Parameters

The possible modes are

Radix: OCTAL, DECIMAL, HEXADECIMAL or BINARY

Display: INSTRUCTION, ASCII, BYTE, WORD or RAD5Ø

Address: SYMBOLIC or NOSYMBOLIC

The default modes are OCTAL, SYMBOLIC and INSTRUCTION.

Display modes INSTRUCTION and ASCII affect length. INSTRUCTION implies that a variable number of words is interpreted. ASCII implies the WORD length. (See Section 7.2.2.)

Modes are interpreted from left to right. If you set the radix, display or address mode more than once in a command, the last setting is used.

Examples

Command	Result
SE M INSTRUCTION	Sets display mode to INSTRUCTION
SE M DECIMAL, BYTE	Sets radix mode to DECIMAL and display mode to BYTE
SET MODE OCTAL, DECIMAL	Sets radix mode to DECIMAL

Related Commands

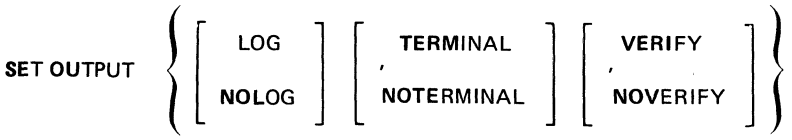
CANCEL MODE

SHOW MODE

SET OUTPUT

The SET OUTPUT command specifies the kind of output MDE/T-11 produces.

Syntax



The shortest form of SET OUTPUT is SE OU

Parameters

LOG turns on the logging mechanism and copies all interactions to the default log file CDS.LOG or to the file named as the log file in the most recent SET LOG command. NOLOG (the initial setting) turns off the logging mechanism.

VERIFY, the initial setting, displays text from a command file on the terminal. NOVERIFY does not display command file text.

TERMINAL displays all MDE/T-11 responses on the terminal. NOTERMINAL does not display MDE/T-11 responses. TERMINAL is the initial setting.

Examples

Command	Result
SE OUT LOG	Turns on logging mechanism
SE OUT NOLOG	Turns off logging mechanism
SE OUT VERI	Displays command file text
SE OUT NOTE	Does not display responses on terminal

Related Command

SET LOG

SET TERMINAL

The SET TERMINAL command tells MDE/T-11 whether you are using a VT100, LA120 or another terminal.

Syntax

SET TERMINAL { LA36 | VT100 | LA120 }

The shortest form of SET TERMINAL is SE TERM

Parameters

VT100 specifies any VT100 series terminal including, optionally, a VT100 with the advanced video option. LA120 specifies the LA120 hard-copy terminal. Use LA36 for all other terminals. An implicit SET TERMINAL command is performed at program start-up.

Examples

Command	Result
SE TERM VT100	Puts MDE/T-11 in VT100 mode, displaying and updating static display and enabling keypad capability
SE TERM LA120	Puts MDE/T-11 in LA120 mode, enabling keypad capability
SE TERM LA36	Sets terminal to hard-copy mode, disabling all special terminal capabilities
SE TERM	Lets MDE/T-11 sense type of terminal being used at program start-up

SET TRACE

The SET TRACE command sets a tracepoint, a point in a program where the execution of the MICRO/T-11 is traced. If a breakpoint exists at the address given in this command, that breakpoint is canceled and the tracepoint is set.

Syntax

SET TRACE [/COUNT : [1] .. 256_{10}] address-expression
[/AFTER : [1] .. 65535_{10}]

The shortest form of SET TRACE is SE T

Parameter

Address-expression must evaluate to an address at which a tracepoint can be set. If a tracepoint exists at the specified address, it is reset with the attributes given in the new command.

Qualifiers

/COUNT:n activates the associated tracepoint every nth time it is signaled. The range of n is 1 to 256.

If /AFTER:count is given, signaling of the tracepoint causes the count to be decremented. If decremented to Ø, the tracepoint is reported; if not, an implicit GO is performed.

/COUNT is hardware-implemented and faster, but limited to 256 (decimal). /AFTER is software-implemented and much slower, but can be as large as 65535 (decimal). Also, the /AFTER counter is displayed with the SHOW TRACE command.

When a tracepoint is activated, MDE/T-11 prints a message indicating the location of the tracepoint and then resumes execution of the target program; that is, an implicit GO is performed.

Examples

Command	Result
SE T START+1Ø	Sets tracepoint at location START+1Ø
SE T /COU:4 START+1Ø	Sets tracepoint at location START+1Ø to be activated every 4th time it is signaled

MDE/T-11 responds to the occurrence of a tracepoint by printing TRACE at PC = address-expression. For example

```
MDE>
TRACE at PC = QUANT
```

Related Commands

CANCEL TRACE

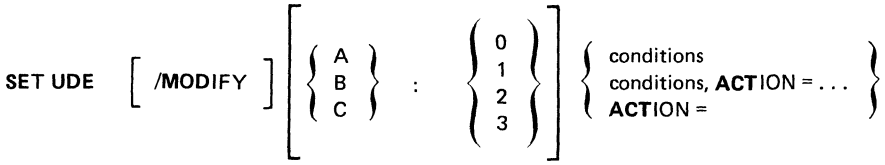
RESET ANALYZER

SHOW TRACE

SET UDE

The SET UDE command establishes conditions and actions for a user-defined event (UDE).

Syntax



The shortest form of SET UDE is SE UDE

Qualifier

/MODIFY lets the command change the setting of an existing event. The board:comparator parameter you give in the command must specify a comparator in which a UDE is set. The command changes only the applicable fields in the associated state template specified in the command.

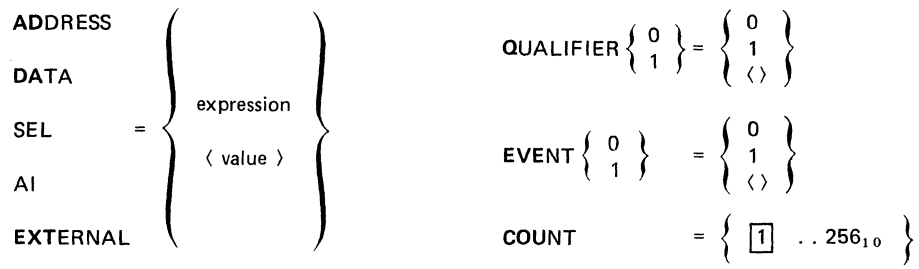
Parameters

Board:comparator specifies the comparator to be associated with the conditions. It has the form

```
{ A } { Ø }
{ B }:{ 1 }
{ C } { 2 }
      { 3 }
```

The letter specifies the state analyzer board, and the number specifies the comparator on that board. The first state analyzer board is A, the second B, and so on. A board comparator of A:1 specifies comparator 1 in analyzer A.

Condition can be any of the following.



ADDRESS = expression

DATA = numeric value

SEL = numeric value

AI = numeric value

EXTERNAL = numeric value (Does not apply if a comparator in the third state analyzer is specified. In external field, probe A applies to state analyzer A only; probe B to state analyzer B only.)

TRANSACTIONS = { READ [DMA] | WRITE [DMA] | FETCH | REFRESH | IACK | ASPI }

ACTIONS can be any of the following.

BREAK	TRACE
TRIGGER	STOP
SIGNAL [QUALIFIER] 0	RESET
SIGNAL [QUALIFIER] 1	

NOTE

The external probe lines are latched on the leading or trailing edge of the clock line on the probe unit. Thus, when the clock line is sampled during a MICRO/T-11 cycle, the latch contains the last value clocked. (See the description of the CONFIGURE CLOCK command in this chapter for information on how to specify whether data on the probe lines is latched on the leading or trailing edge of the clock signal.

EVENT = numeric value (see subsection 3.2.1.2)

QUALIFIER = numeric value (see subsection 3.2.1.2)

COUNT = expression

You can enter an expression or a single numeric value into a state template. If you enter an expression, the value of the expression sets the appropriate field.

You can enter a numeric value by placing it within angle brackets (<>). You can further specify the radix of the value with a radix override (% followed by the first letter of the radix and the numeric value enclosed in single quotes). For example, you would enter a binary 3 into a UDE as follows.

```
DATA = <%B'XXXXXXXX00000011'>
```

You can also enter a numeric value by simply enclosing it within angle brackets. In this case, the radix will be interpreted as the radix mode currently set.

X specifies a don't-care. For example, suppose the current mode is octal and the following state template expression is given.

```
<7XXX>
```

In this case, bits Ø through 8 are don't-cares; bits 9, 1Ø and 11 are 1s; bits 12 through 15 are Øs.

You can use a radix override (described in Chapter 7) to specify digit masks in binary, octal or hexadecimal.

You set the frequency with which the event is signaled by means of the /COUNT qualifier.

ACTION specifies the actions that are taken if the event described in the command is signaled. Possible actions are

BREAK — Stop execution of target program.

TRIGGER — Generate output pulse on MDE/T-11 external TRIGGER connector.

TRACE — Save data pattern for each successive bus cycle in trace RAM, starting with next bus cycle.

STOP — Terminate data pattern capture in trace RAM after including data pattern for next bus cycle.

RESET — Clear all EVENT and QUALIFIER bits, and reset all COUNT counters on all state analyzers (reset action does not reset AFTER counters on breakpoints, tracepoints and watchpoints).

SIGNAL [QUALIFIER] Ø
Set qualifier bit Ø on all state analyzers.

SIGNAL [QUALIFIER] 1
Set qualifier bit 1 on all state analyzers.

The behavior of TRACE and STOP is determined by the way tracing was specified in the CONFIGURE ANALYZER command.

NOTE

Activating an event in comparators Ø or 1 on a given state analyzer implicitly sets event flags Ø and 1.

Examples (assume OCTAL radix mode)

Command	Result
SE UDE A:1 AD=START+2Ø,ACT=BR	Loads state template, consisting of address field with value START+2Ø and all other fields with ignored values, into comparator 1 in analyzer A. Assumes count of Ø. If template is matched, event is activated and BREAK is executed, stopping program execution. Because event is set in comparator 1, occurrence of event implicitly sets EVENT bit 1 on analyzer A.

SE UDE A:2 DA=2Ø,ACT=TRI
Loads state template, consisting of data field of 2Ø and all other fields of ignored values, into comparator 2 in analyzer A. If template is matched, event is activated and TRIGGER is executed, triggering external signal.

SE UDE B:Ø AD=START+2Ø,AI=1,TRAN=FETCH,ACT=TRA
Loads state template, with ADDRESS, AI and TRANSACTION fields set, into comparator Ø in analyzer B. If template is matched, event is activated and TRACE is executed, capturing bus cycles in trace RAM from next bus cycle on. Because event is set in comparator Ø, occurrence of event implicitly sets EVENT bit Ø on analyzer B.

SET UDE /MODIFY B:Ø ADDRESS=START+22
Changes address field of state template associated with comparator B:Ø to START+22

SET UDE A:3 ADDRESS=LOOP,DATA=<XXXXX4>,ACTION = STOP RESET
If MICRO/T-11 cycle, with ADDRESS field equal LOOP and low-order three bits of DATA field equal 4, is matched, executes STOP and RESET. STOP turns off bus cycle tracing after next bus cycle; RESET resets flags associated with previous events.

MDE/T-11 responds to the occurrence of a UDE by printing the user-defined event ordinal. For example

```
MDE>
UDE #2
```

Related Commands

CANCEL UDE

RESET ANALYZER

SHOW UDE

SET WATCH

The SET WATCH command sets a watchpoint, which specifies a memory location to be watched. If the MICRO/T-11 performs a write to the specified location, the watchpoint is signaled, processor execution halts, and a watchpoint message is displayed. If a watchpoint exists at the specified address, it is reset with the attributes given in the new command.

Syntax

SET WATCH [**/COUNT** : [1] .. 256_{10} | **/AFTER** : [1] .. 65535_{10}] address-expression

The shortest form of SET WATCH is SE W

Parameter

Address-expression is a location in memory.

Qualifiers

/COUNT:n activates the associated watchpoint the nth time it is signaled.

If /AFTER:n is given, signaling of the watchpoint causes the count to be decremented. If decremented to Ø, the watchpoint is reported; if not, an implicit GO is performed.

/COUNT is hardware-implemented and faster, but limited to 256 (decimal). /AFTER is software-implemented and much slower, but can be as large as 65535 (decimal). Also, the /AFTER counter is displayed with the SHOW WATCH command. You can use the /COUNT and /AFTER qualifiers together in this command, extending the event count for watchpoints to 16,776,96Ø.

When a watchpoint is activated, MDE/T-11 displays a message indicating the old and new contents of memory.

Examples

Command	Result
SE W SUM1	Sets watchpoint at memory location SUM1
SE W /COUNT:3 SUM1	Sets watchpoint at memory location SUM1 to be activated when contents of SUM1 have changed three times

When a watchpoint occurs, MDE/T-11 displays a message in the following format.

Watch at PC=address-expression; value was: old-expression, is now: new-expression; current PC = n

For example

```
WATCH at SUM1; value was: 5, is now: 1Ø; current PC = START+3Ø
```

Related Commands

CANCEL WATCH

RESET ANALYZER

SHOW WATCH

SHOW ALL

The SHOW ALL command displays all events including the ordinal identifying each event, the value of the counts associated with each event by the /COUNT and /AFTER qualifiers, the address at which each event is set, and the event type.

Syntax

SHOW ALL

The shortest form of SHOW ALL is SH ALL

Examples

Command	Result
SH ALL	Displays status of each type of event

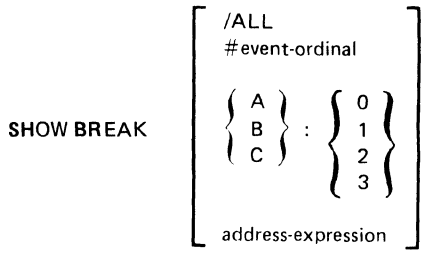
A report like the following is produced.

```
MDE>SH ALL<RET>
Ø: A:Ø BREAKpoint /COUNT:1 /AFTER:1 (1 left) Address = START+1Ø
1: A:1 TRACEpoint /COUNT:5 /AFTER:5 (3 left) Address = SUM1
2: A:2 WATCHpoint /COUNT:1 /AFTER:1 (1 left) Address = QUANT
MDE>
```

SHOW BREAK

The SHOW BREAK command produces a report on the current status of breakpoints. This report gives the ordinal number assigned to each breakpoint, the value of the counts associated with each breakpoint by the /COUNT and /AFTER qualifiers, the state analyzer and comparator in which the breakpoint is set, and the address at which the breakpoint is set.

Syntax



The shortest form of SHOW BREAK is SH BR

Qualifier

/ALL (or no qualifier or parameter) displays the status of all breakpoints.

Parameters

#event-ordinal is an identification number from Ø through 11 preceded by a number sign, assigned to the breakpoint by MDE/T-11.

Board:comparator specifies the state analyzer (A, B or C) and comparator (Ø, 1, 2 or 3) used to set the breakpoint.

Examples

Command	Result
SH BR/ALL	Displays status of all breakpoints

For each breakpoint, MDE/T-11 gives the following information.

Ordinal associated with breakpoint and comparator used

Value of counts associated with breakpoint by /COUNT and /AFTER qualifiers

Location of breakpoint

For example

```
MDE>SH BR<RET>
#3 A:3 Breakpoint /COUNT:3 /AFTER:5 (3 left) Address = START+1Ø
```

Related Commands

CANCEL BREAK

SET BREAK

SHOW CONFIGURE

The SHOW CONFIGURE command displays the current configuration of the emulator, memory simulator, and state analyzers.

Syntax

SHOW CONFIGURE

The shortest form of SHOW CONFIGURE is SH CON

Examples

Command	Result
SH CON	Produces report on configuration

The following is an example of a report produced by this command.

```
Mode Register:      Read from EMULATOR
Mode Settings:      NORMAL, STANDARD, STATIC, PROCESSOR, USER
                    Start address = 140000, Bus Width = 16 bits

Processor Clock:    EXTERNAL

Fetch Timeout:      DISABLED
State Analyzer:     Clock A+; AI = Row; Trace AFTER /RETAIN:10

Memory Map (octal):

   From         To     Where      Write Prot?

      0      35776     Absent     NA
  36000      50376     Sim        No
  50400     177776     Absent     NA
```

Related Commands

CONFIGURE ANALYZER

CONFIGURE MEMORY

CONFIGURE MODE

SHOW DEFINE

The SHOW DEFINE command displays the values assigned to a defined symbol.

Syntax

SHOW DEFINE { /ALL | symbol-name }

The shortest form of SHOW DEFINE is SH DEF

Parameter

Symbol-name prints the value associated with that symbol.

Qualifier

/ALL displays information about all defined symbols.

Examples

Command	Result
SH DEF FETCH	Displays value associated with FETCH
SH DEF/ALL	Displays values associated with all defined symbols

The report produced by /ALL lists each defined symbol and its value in the current radix. For example

```
MDE>SH DEF/ALL<RET>
          Symbol      Value
          ELOOP       40044
          MAX         40004
```

Related Commands

DEFINE

UNDEFINE

SHOW KEYDEFINE

The SHOW KEYDEFINE command displays your definitions for the keypad PF keys.

Syntax

```
SHOW KEYDEFINE
```

The shortest form of SHOW KEYDEFINE is SH K

Examples

This command produces a display for the defined keys in the form

```
PF1:  'command string' 'command string' ...
PF2:  'command string' 'command string' ...
PF3:  'command string' 'command string' ...
PF4:  'command string' 'command string' ...
```

If a key is not defined, it does not appear in the display.

Related Command

KEYDEFINE

SHOW MODE

The SHOW MODE command produces a report on the current radix, display and address modes.

Syntax

SHOW MODE

The shortest form of SHOW MODE is SH M

Examples

Command	Result
SH M	Displays current mode settings

This command produced the following report.

The current modes are: OCTAL,WORD,SYMBOLIC

Related Commands

CANCEL MODE

SET MODE

SHOW OUTPUT

The SHOW OUTPUT command produces a report on the devices being used for output.

Syntax

SHOW **OU**TPUT

The shortest form of SHOW OUTPUT is SH OU

Examples

Command	Result
SH OUT	Displays current output settings

This command produces a report in the form

```
output:  [no]verify, [no]terminal, and [no]logging (Log file is
         'file-spec')
```

For example

```
MDE> SH OUTPUT<RET>
output:  VERIFY,  TERMINAL, and LOGGING (Log file is 'CALC.LOG')
```

Related Commands

SET LOG

SET OUTPUT

SHOW SYMBOL

The SHOW SYMBOL command displays the value of a program symbol or the values of all program symbols.

Syntax

SHOW SYMBOL { /ALL | symbol-name }

The shortest form of SHOW SYMBOL is SH SYM

Parameter

Symbol-name displays the value associated with that symbol.

Qualifier

/ALL displays the values for all symbol names.

Examples

Command	Result
SH SYM START	Displays value associated with START

The report produced by /ALL lists each symbol and its value in the current radix. For example

```
MDE>SH SYM/ALL<RET>
          Symbol      Value
          QUANT       40070
          START       40024
          SUM1        40020
          SUM2        40022
          V           40000
MDE>
```

SHOW TARGET

The SHOW TARGET command displays a one-line report showing whether the MICRO/T-11 is powered up, running or in the pause state, and the source of interrupts you specified in a SIGNAL command.

Syntax

SHOW **TA**RGET

The shortest form of SHOW TARGET is SH TA

Examples

Command	Result
SH TA	Displays target status

The following is an example of a SHOW TARGET report.

```
Processor is powered up, halted;  interrupts taken from target
```

Related Commands

GO

HALT

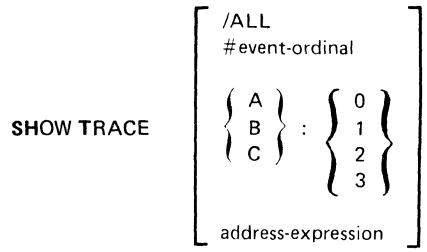
POWER

SIGNAL

SHOW TRACE

The SHOW TRACE command produces a report on the current status of tracepoints. This report gives the ordinal number assigned to each tracepoint, the value of the counts associated with each tracepoint by the /COUNT and /AFTER qualifiers, the state analyzer and comparator in which the tracepoint is set, and the address at which the tracepoint is set.

Syntax



The shortest form of SHOW TRACE is SH T

Qualifier

/ALL (or no qualifier or parameter) displays the status of all tracepoints.

Parameters

#event-ordinal is an identification number from Ø through 11 preceded by a number sign, assigned to the tracepoint by MDE/T-11.

Board:comparator specifies the state analyzer (A, B or C) and comparator (Ø, 1, 2 or 3) used to set the tracepoint.

Examples

Command	Result
SH T	Displays status of tracepoints

The report lists all tracepoints set in the program. For each tracepoint, the following information is given.

Ordinal associated with tracepoint and comparator used

Value of counts assigned to tracepoint by /COUNT and /AFTER qualifiers

Location of tracepoint

For example

```
MDE>SH TRA A:3<RET>
#3 A:3 TRACEpoint /COUNT:5 /AFTER:1 (1 left) Address = QUANT
```

Related Commands

CANCEL TRACE

SET TRACE

SHOW UDE

The SHOW UDE command produces a report on the current status of user-defined events. This report gives the ordinal number, the state template, the board and comparator, and the count value for each event.

Syntax

SHOW UDE [/ALL | #event-ordinal | { A | B | C } : { 0 | 1 | 2 | 3 }]

The shortest form of SHOW UDE is SH UDE

Parameters

Board:comparator specifies the state analyzer (A, B or C) and comparator (Ø, 1, 2 or 3) used to set the event.

#event-ordinal is an identification number from Ø through 11 preceded by a number sign, assigned to the event by MDE/T-11.

Qualifier

/ALL (or no qualifier or parameter) displays all user-defined events.

Examples

Command	Result
SH UDE	Displays status of each user-defined event

The report lists all events set in the program. For each event, the following information is given.

Ordinal associated with event

Value of count associated with event by /COUNT qualifier

Contents of comparator (state template). Only fields not solely consisting of don't-cares (ignored bits) are shown.

The form of a report is

```
#ordinal  User Defined Event board:comparator
     TRANSACTION = name
     ADDRESS     = bbbbbbbbbbbbbbbb = nnnnnn
     DATA        = bbbbbbbbbbbbbbbb = nnnnnn
     SEL         = bb
     EXTERNAL    = bbbbbbbb
     AI          = bbbbbbbb
     EVENT Ø     = b
     EVENT 1     = b
     QUALIFIER Ø = b
     QUALIFIER 1 = b
     COUNT       = count value
```

where bbb ... = the binary mask in the given state template field.

nnn ... = the binary mask (with Xs interpreted as Øs) translated into the current radix.

The ordinal and count are printed in decimal regardless of the current radix. The character string indicates whether a bit is being tested; X means masked and 1 means tested.

Suppose the following user defined event is set.

```
MDE>SET UDE A:2 TRANS=FETCH,DATA=<4XXX>,ACTION=STOP TRACE<RET>
```

The command SHOW UDE produces the following report.

```
#2 User Defined Event
     Transaction = Fetch
     Data        = ØØØØ1ØØXXXXXXXXX = 4ØØØ
     Action      = STOP,TRACE
     Count       = 1
```

Related Commands

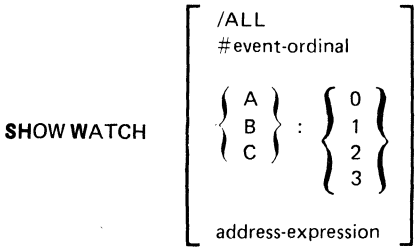
CANCEL UDE

SET UDE

SHOW WATCH

The SHOW WATCH command produces a report on the current status of watchpoints. This report gives the ordinal number assigned to each watchpoint, the value of the counts associated with each watchpoint by the /COUNT and /AFTER qualifiers, the state analyzer and comparator in which the watchpoint is set, and the address at which the watchpoint is set.

Syntax



The shortest form of SHOW WATCH is SH W

Qualifier

/ALL (or no qualifier or parameter) displays the status of all watchpoints.

Parameters

#event-ordinal is an identification number from Ø through 11 preceded by a number sign, assigned to the watchpoint by MDE/T-11.

Board:comparator specifies the state analyzer (A, B or C) and comparator (Ø, 1, 2 or 3) used to set the watchpoint.

Examples

Command	Result
SH W	Displays status of watchpoints

The form of a line in a report is

```
#ordinal:  comparator WATCHpoint /COUNT:count /AFTER:n (n left)
           Address = address
```

The ordinal and count are printed in decimal regardless of the current mode. For example

```
MDE>SH WAT #1
#1 A:1 WATCHpoint /COUNT:1 /AFTER:5 (3 left) Address = SUM1
```

Related Commands

CANCEL WATCH

SET WATCH

SIGNAL

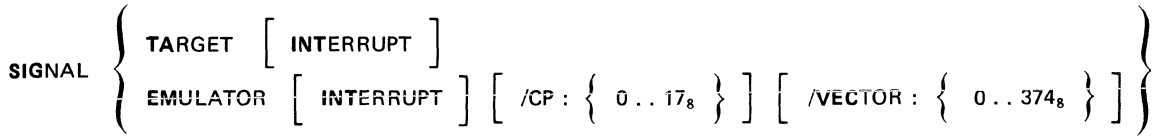
The SIGNAL command selects the target hardware or the MDE/T-11 emulator as the source of interrupts. When simulating interrupts, command parameters let you specify the CPU interrupt priority level and interrupt vector address.

NOTE

DMA requests are ignored whenever interrupt simulation is in effect. However, power fail and halt traps issued by target hardware are executed.

Using the PDP-11 WAIT instruction causes the MICRO/T-11 to hang. In this situation, MDE/T-11 cannot assume control of the MICRO/T-11 and cannot, therefore, control the MICRO/T-11 to perform some debugging functions. You can avoid this problem by using a BR. instruction. The BR. instruction duplicates the interrupt idling obtained with the WAIT instruction.

Syntax



The shortest form of SIGNAL is SIG

Parameters

TARGET specifies the target as the source of interrupts. EMULATOR specifies the emulator as the source of interrupts.

Qualifiers

If you do not give a value for /CP, no interrupt is generated. If you give a value for /CP, that value is presented to the MICRO/T-11 on the CP<3> through CP<Ø> lines.

/VECTOR:n is any multiple of 4 in the range Ø through 374 (octal). If you do not give a value for /VECTOR, the MICRO/T-11 gets the vector address from an internal fixed table by decoding the inputs HALT, PF, CP<3:Ø>. If you give a value, it is presented to the MICRO/T-11 DAL<7:2>. Do not use bits 1 and Ø, because the vector address must be a multiple of 4.

Examples

Command	Result
SIG TA	Sets target as source of interrupts
SIG EMU	Sets emulator as source of interrupts
SIG EMU CP:7	Asserts CP<3> = Ø and CP<2:Ø> = 1; uses default MICRO/T-11 vector for interrupt (octal 114)
SIG EMU CP:1 VEC:1ØØ	Asserts CP<3:1> = 1 and CP<Ø> = Ø; presents vector = 1ØØ to MICRO/T-11 during resulting IACK cycle

Related Command

SHOW TARGET

STEP

The STEP command executes the target program a specified number of instructions at a time.

Syntax

STEP [increment]

The shortest form of STEP is S

Parameter

Increment is an expression that evaluates to an unsigned 16-bit value. It specifies how many instructions to step through before halting. If you do not specify increment, MDE/T-11 stops after one instruction.

MDE/T-11 single steps the program as many times as indicated by increment. Since single-stepping alters the performance of the program, do not use this process if real-time execution is desired.

Examples

Command	Result
S	Steps one instruction
S 5	Steps five instructions then halts

MDE/T-11 responds by displaying the registers. The following is an example for a non-VT100 terminal.

```
MDE>S<RET>
;MDE-I-PROSTETO, Processor single stepped to PC=ELOOP+12 : MOV@#100,R0

R0:       40004
R1:           7
R2:           3
R3:      177777
R4:      177777
R5:      177777
SP:       40000
PC:    ELOOP+12
PS:         340
MDE>
```

For VT100 terminals, the registers are shown in the static display region of the screen.

Related Command

GO

STOP

The STOP command lets you leave MDE/T-11 and return to the VAX/VMS operating system. MDE/T-11 closes any files that are open (see description of EXIT command).

Syntax

```
STOP
```

The shortest form of STOP is STO

Example

Command	Result
STO	Closes files and returns to monitor

Related Command

EXIT

UNDEFINE

The UNDEFINE command removes defined symbols from the symbol table, symbols you created with the DEFINE command.

Syntax

UNDEFINE symbol [,symbol,...]

The shortest form of UNDEFINE is UND

Parameter

Symbol is any symbol defined by the DEFINE command. Redefined symbols loaded from the program symbol table return to their initial (loaded) values. MDE/T-11 cannot remove loaded symbols from the program symbol table.

Examples

Command	Result
UND ALPHA	Removes symbol ALPHA
UND ARG1,TEMP	Removes symbols ARG1 and TEMP

Related Commands

DEFINE

SHOW DEFINE

WAIT

The WAIT command suspends command input until the MICRO/T-11 emulator is placed in the pause state. This command suspends input from the terminal or from an indirect command file. Command input is resumed when the MICRO/T-11 enters the pause state for any reason. You can abort this command by typing CTRL/C twice.

Syntax

```
WAIT
```

The shortest form of WAIT is WAI

Example

The following commands begin program execution at label START, continue until just prior to execution of the instruction at label LOOP, and perform other MDE/T-11 commands thereafter.

```
SET BREAK LOOP
GO START
WAIT
  .
  .
  .
```

You can place this sequence in an indirect command file, or assign it to a keypad PF key.

CHAPTER 9

MESSAGES

This chapter describes MDE/T-11 messages that you may encounter when developing MICRO/T-11 applications. It explains the general format of MDE/T-11 messages and provides an alphabetical list of messages by message type.

9.1 MESSAGE FORMAT

All messages appear in the following form.

```
sMDE-X-MESSAGE TEXT
```

The sMDE- field identifies an MDE/T-11 success, information, warning or error message (rather than a VAX/VMS system message) as follows.

Identifier	General Type
;MDE-	Success, information
%MDE-	Warning
?MDE-	Error (severe, internal, fatal)

X- is one of six possible message codes (Table 9-1) indicating successful completion or the severity of an error.

Following the message code letter, the message type is displayed in uppercase letters as an aid for quick identification. The length of the message type is nine characters maximum.

The text portion of the message provides more explicit information as an aid for error recovery or corrective action. For example, the text may contain information about the particular construct causing an error.

```
Unable to cancel activated BREAKpoint 'name'
```

The quoted string 'name' identifies the breakpoint. For example

```
Unable to cancel activated BREAKpoint START+2Ø
```

The remaining sections in this chapter list MDE/T-11 messages alphabetically by type.

Table 9-1
MDE/T-11 Message Codes

Level	Message Code	Effect
Success	S	MDE/T-11 successfully completed the specified operation.
Information	I	MDE/T-11 detected a condition that may require your attention or some action.
Warning	W	MDE/T-11 detected a condition that may cause errors in execution. Corrective action may be necessary.
Severe error	E	MDE/T-11 command execution failed. This is generally the result of a serious error that prevented execution.
Internal error	C	An internal MDE/T-11 error occurred preventing normal operation. The error may be the result of a hardware or software condition that requires the restarting of the MDE/T-11 system.
Fatal error	F	A fatal error occurred external to MDE/T-11 software. MDE/T-11 exits and operator control returns to the VAX/VMS command level.

9.2 SUCCESS MESSAGES

The following success messages report that an operation was successful. MDE/T-11 gives success verification on just a few commands. You may assume that any command executed without messages was successful.

;MDE-S-LOAGOODLD, Load complete

The LOAD command was successful.

;MDE-S-PROGSTART, Program started at PC = ,nnnnnn

The GO command was successful. The PC at the time the processor started is given.

9.3 INFORMATION MESSAGES

The following information messages inform you of certain (normal) actions taken as a result of a command.

;MDE-I-COPDATLOS, Data in destination lost

A COPY operation was completed, but it destroyed data in configured memory.

;MDE-I-EMUCHKPOD, Check POD switches, repeat power up sequence

The MICRO/T-11 cannot be started. It may have no clock signal. Check the setting of the MDE/T-11 pod clock switches.

;MDE-I-KDFNONDEF, None are defined

A KEYDEFINE/CLEAR/ALL was given, and no keypad keys were defined.

;MDE-I-LOANOBIN, No binary file found

You issued a LOAD command that contained an unquoted file specification with no extension. Therefore the binary and symbol files were loaded but the binary file was not found.

;MDE-I-LOANOSYM, No symbol file found

LOAD was given with an unquoted file specification and no extension. Therefore the binary and symbol files were loaded but the symbol file was not found.

;MDE-I-NOXFRADDR, No transfer address

A binary file (memory image) loaded successfully, but it did not specify a transfer address.

;MDE-I-PROHALAT, Processor execution halted at PC = base + offset: instruction

You issued a HALT command, and the MICRO/T-11 halted your application program at the address of the current PC.

;MDE-I-PROSTETO, Processor execution started at PC = base + offset: instruction

You issued a GO command, and the MICRO/T-11 began to execute your application program at the address of the current PC.

;MDE-I-REPVFYFAL, Verification failed at address nnnnn

You specified /VERIFY in a LOAD or DEPOSIT command, but the verification read-back failed at the address displayed.

;MDE-I-SHWNOSYMB, No symbols defined

You issued a SHOW DEFINE or SHOW SYMBOL command, but no symbols were defined.

;MDE-I-STEPABORT, Single stepping aborted on iteration n

The STEP command you issued was aborted due to the occurrence of an event. The number of instructions successfully stepped is reported.

;MDE-I-TIMOUT, Emulator fetch timout

Fetch timeout occurred in response to the CONFIGURE TIMEOUT ENABLED you issued.

;MDE-I-TRMKEYACT, Keypad is now active

MDE/T-11 detected that an LA12Ø terminal is being used as the console terminal. The LA12Ø keypad was activated.

9.4 WARNING MESSAGES

The following messages are warnings, indicating an MDE/T-11 operation you specified was completed but some abnormal result occurred. Warning messages can also indicate some side effect has occurred as a result of normal action.

%MDE-W-COPOVRLAP, Destination overlaps source

You issued a COPY command in which the destination block overlaps the source block. An undesirable side effect may have occurred.

%MDE-W-DEPREGBYT, /WORD assumed in register deposit

You attempted to DEPOSIT in one or more registers when in the BYTE display mode or with the /BYTE qualifier in the command. The deposit was executed in WORD mode.

%MDE-W-EMUADRFRC, Target address space read/write forced to word boundary

You specified an odd address in a DEPOSIT or EXAMINE command, requiring MDE/T-11 to adjust to the word address.

%MDE-W-EMUVFYFAL, Verification error on load or deposit

You specified /VERIFY in a LOAD or DEPOSIT command, and verification read-back failed.

%MDE-W-INIEMPBUS, Empty bus

MDE/T-11 hardware is improperly configured.

%MDE-W-KDFXSSTRS, n excess strings ignored

The KEYDEFINE commands you issued specified more than eight commands. The extra KEYDEFINE commands were ignored.

%MDE-W-LOAABOLOA, LOAD aborted via ^C

You typed two CTRL/Cs while loading was in progress.

%MDE-W-LOAXFRADD, Transfer Address nnnnnn, not loaded

MDE/T-11 could not load the target PC with the program's transfer address in response to the LOAD command you issued, because the target processor was not powered up.

%MDE-W-LOGOUTERR, Log file output error, file closed

The log file currently in use reached its preallocated size; thus it was closed.

%MDE-W-NOLDXFRAD, Could not load transfer address

MDE/T-11 could not load the target PC with the program's transfer address. This message normally precedes the %MDE-W-LOAXFRADD message.

9.5 ERROR MESSAGES

Error messages can be severe, internal or fatal.

9.5.1 Severe Error Messages

The following severe error messages indicate an error in constructing a command or in using MDE/T-11.

?MDE-E-ARGOUTRNG, Argument out of range

You specified a numerical argument that was too large or too small.

?MDE-E-BADBUS, System on bus m not legally configured

MDE/T-11 detected an improperly configured backplane. More than one emulator is present, or the device number configuration is improper.

?MDE-E-BADEMU, Emulator on bus m failed diagnostics

MDE/T-11 detected a hardware error in the emulator. Check it before attempting to debug.

?MDE-E-BADEVTBD, State Analyzer n on bus m failed diagnostics

MDE/T-11 detected a hardware error in a state analyzer. Check it before attempting to debug.

?MDE-E-BADLOAD, Could not load

The LOAD command you issued failed. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-BADMEMSIM, Memory simulator n on bus m failed diagnostics

MDE/T-11 detected a hardware error in a memory simulator. Check it before attempting to use MDE/T-11 debugging commands.

?MDE-E-CLREVTFAL, Event could not be cleared

MDE/T-11 could not clear an event comparator. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-COMPLEX, Syntax, probably expression, too complex

You probably issued a command containing an expression that had too many levels of nested parentheses.

?MDE-E-COPNOTCON, Memory not configured or not contiguous

MDE/T-11 detected an attempt to reference memory not configured with the CONFIGURE MEMORY command.

?MDE-E-COPNOTCOP, No copy performed

The COPY command you issued failed. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-DEPARGCNT, Too much or too little data for deposit

You issued a DEPOSIT command to deposit in registers but the number of data elements did not match the number of registers, or to deposit in a memory range with more data than the size of the range you specified permits.

?MDE-E-DEPBADRNG, Illegal deposit range specified

The range you specified in a DEPOSIT command had a bottom address greater than the top address; for example, 200:100.

?MDE-E-EMUBADHRD, No processor clock, cannot run DCT11

MDE/T-11 detected an attempt to manipulate the MICRO/T-11 processor, or to examine or deposit target memory or registers, but the MICRO/T-11 had no clock signal. Check the setting of the MDE/T-11 pod clock switches.

?MDE-E-EMUBADPWR, Wrong power state for action

MDE/T-11 detected an attempt to manipulate target memory or registers before powering up the MICRO/T-11. You must first power up the target hardware.

?MDE-E-EMUINTFAL, T11 failed to take interrupt (check processor priority)

The SIGNAL INTERRUPT command you issued signaled an interrupt, but the MICRO/T-11 did not acknowledge it. The CPU priority (in the PS) probably is too high.

?MDE-E-EVEQUAL, EVENT or QUALIFIER number must be 0 or 1

You incorrectly specified one or more event or qualifier flags in a SET UDE command.

?MDE-E-EVTBADCMP, Comparator out of range

The comparator you specified in a SET UDE command does not exist.

?MDE-E-EVTBADCNT, Count value out of range

You attempted to set an event using a count value greater than 256.

?MDE-E-EVTCMPUSD, Comparator specified already in use

You attempted to set a UDE using a comparator already in use.

?MDE-E-EVTNOTSET, Event could not be set

The SET BREAK, SET TRACE, SET WATCH or SET UDE command you issued failed. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-EXCESSDET, Excess State Analyzers ignored

You attempted to configure more than three state analyzers.

?MDE-E-ILLRANGE, range given for memory mapping is not ascending

The memory range you specified in a CONFIGURE MEMORY command is illegal. The bottom address is greater than the top address.

?MDE-E-INIDEVGAP, Noncontiguous device numbers on bus

This indicates an improperly configured MDE/T-11 system. Check the MDE/T-11 hardware installation.

?MDE-E-INIMULEMU, Multiple Emulators on bus

This indicates an improperly configured MDE/T-11 system. Check the MDE/T-11 hardware installation.

?MDE-E-ININOEDET, No event detector on bus

This indicates an improperly configured MDE/T-11 system. Check the MDE/T-11 hardware installation.

?MDE-E-ININOMSIM, No Memory Simulator on bus

This indicates an improperly configured MDE/T-11 system. Check the MDE/T-11 hardware installation.

?MDE-E-ININOTEMU, No target Emulator on bus

This indicates an improperly configured MDE/T-11 system. Check the MDE/T-11 hardware installation.

?MDE-E-INSTFMT, MACRO-11 assembly error

The assembly of MACRO-11 code failed. The offending line and a pointer to the syntax error precedes the message.

?MDE-E-INSTODD, Attempt to deposit MACRO-11 to odd address

The DEPOSIT command you issued specified MACRO-11 instructions to be deposited to an odd address.

?MDE-E-INSTUSE, Use of MACRO-11 in register or range deposit

You attempted to deposit MACRO-11 code into registers.

?MDE-E-KDFILLKEY, Illegal or undefined key

You attempted to define a nonexistent key.

?MDE-E-KDFKEYDEF, Key PFn already defined

You attempted to define a keypad key that is already defined. First use the KEYDEFINE/CLEAR command to clear the key.

?MDE-E-KDFNOT100, Terminal must be a VT100 or LA120

You attempted to define keys on a terminal with an unsupported keypad. Only LA120 and VT100 keypads are supported by MDE/T-11.

?MDE-E-LEXERR, Syntax error at or near "xxx"

The command syntax you issued contains one or more lexical errors. Probably the numbers you specified are not in the current radix, or you made an error in constructing a radix override (for example, %O'xxx').

?MDE-E-LOAFILCON, Conflicting file type switches specified

You issued a LOAD command in which /SYMBOL and /BINARY were specified. Only one can be specified.

?MDE-E-LOAHASHDR, Task Image has header; rebuild with /-HD

The memory image file built with the RSX-11 task builder is incorrect (contains a header).

?MDE-E-LOAHRDERR, Hardware error, load aborted

MDE/T-11 could not load target or simulator memory. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-LOAILLFMT, Bad binary file format, binary load aborted

The memory image file you specified in a LOAD command was not in the format implied by its name or by the file type switch.

?MDE-E-LOANOFILE, File not found or could not be opened

The file you specified in a LOAD command could not be found or opened.

?MDE-E-LOAREAERR, Read error, load aborted

A read error occurred while reading a binary or symbol file during a load operation.

?MDE-E-LOARESOVR, Task Image has memory-resident overlays

The memory image file built with the RSX-11 task builder is incorrect (contains memory resident overlays).

?MDE-E-LOASYMFMT, Bad symbol file format, symbol load aborted

The symbol file you specified in a LOAD command has an incorrect internal format.

?MDE-E-LOAUNKFMT, Unknown file format; must use /BINARY or /SYMBOL

You included a quoted file specification, or a file specification with a nonstandard extension, in a LOAD command without a file type qualifier.

?MDE-E-LOGTOOLNG, Log file name exceeds 50 characters

The log file name you specified in a SET LOG command is longer than 50 characters.

?MDE-E-MEMOUTMEM, Insufficient physical memory

You attempted to configure memory, but no free 8KB simulator memory blocks are available.

?MDE-E-MEMNOTCON, Memory is not configured

The application program attempted to reference memory not configured by the CONFIGURE MEMORY command.

?MDE-E-MEMREAVIO, Memory read violation, PC = nnnnnn

The application program attempted to access (read) memory configured as read-protected and then halted.

?MDE-E-MEMWRIVIO, Memory write violation, PC = nnnn

The application program attempted to write to memory configured as read-only and then halted.

?MDE-E-MICNOTRUN, Processor is not running

You attempted to halt the MICRO/T-11 when it was in the pause state.

?MDE-E-MICWASRUN, Processor is running

You issued a GO or STEP command during application program execution.

?MDE-E-NOCHAN, No free IO channels

No channels are available for opening a file. This can happen when using indirect command files or when attempting to start logging.

?MDE-E-NOCLOCK, Clock configure failed

MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-NODEPOSIT, Cannot deposit

The DEPOSIT command you issued failed. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-NOEVENT, No such event set

You attempted to cancel an event that was not set.

?MDE-E-NOEVTCLRD, No event(s) cleared

MDE/T-11 could not clear an event. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-NOEXAMINE, Unable to examine location

The EXAMINE command you issued failed. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-NOEXREGS, Cannot examine registers

A register EXAMINE command failed. The MICRO/T-11 probably has no clock signal. Check the setting of the MDE/T-11 pod clock switches.

?MDE-E-NOFILE, Missing file specification

You did not include a required file specification in a command.

?MDE-E-NOFREECOMP, No free comparator for event

You attempted to set an event when no event comparators were available on the specified state analyzer. First cancel an event on that analyzer or use another analyzer.

?MDE-E-NOGO, Could not start at current PC

MDE/T-11 could not start the MICRO/T-11.

?MDE-E-NOHLPCHAN, No free channel for help i/o

No channels are free for opening the help file. This occurs when a HELP command is issued within a command file.

?MDE-E-NOHLPFILE, Could not find HELP file

MDE/T-11 could not find the help file.

?MDE-E-NOINIT, Hardware was not successfully initialized

Some error occurred that caused the initialization process to fail. MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-NOMEMCON, Memory configure failed

?MDE-E-NOMEMSIZ, Memory size configure failed

MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-NOMODEREG, Mode register setting failed

MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-NONEXHRDW, Non-existent hardware specified in command

You attempted to set an event using a state analyzer that does not exist.

?MDE-E-NOOPEN, Cannot open file

The file you specified cannot be found, or file access failed.

?MDE-E-NOPOWER, Power configure failed

The MICRO/T-11 probably has no clock signal. Check the setting of the MDE/T-11 pod clock switches.

?MDE-E-NOSETUP, Emulator setup failed

MDE/T-11 usually precedes this message with another identifying the source of the error.

?MDE-E-NOSUCHEVT, No such event is set

You attempted to clear an event that was not set.

?MDE-E-NOSUCHSYM, Symbol "xxx" does not exist

The symbol you specified in a command was not loaded from a program symbol table or defined by the DEFINE command.

?MDE-E-ODDADDR, Illegal use of odd address

You specified an odd address in an EXAMINE or DEPOSIT command, but MDE/T-11 was not in the BYTE display mode or you did not include the /BYTE qualifier.

?MDE-E-PCRDFAIL, Could not read PC

MDE/T-11 could not read the target PC. The MICRO/T-11 probably has no clock signal. Check the setting of the MDE/T-11 pod clock switches.

?MDE-E-RANGE, Invalid range

You specified an address range in an EXAMINE or DEPOSIT command with a bottom address greater than the top address (for example, 200:100).

?MDE-E-READERR, File read error

An attempt to read a previously opened file failed.

?MDE-E-SIGINTNAK, Interrupt not acknowledged by processor

The SIGNAL INTERRUPT command you issued signaled an interrupt, but the processor did not acknowledge it. The CPU priority (in the PS) probably is too high.

?MDE-E-SIGXSARGS, Excess arguments supplied for SIGNAL command

You included too many parameters in a SIGNAL command.

?MDE-E-STEPFAIL, Single stepping failed on iteration: n

The MICRO/T-11 could not be single stepped as you specified in a STEP command. N is the number of instructions that were successfully executed.

?MDE-E-SYNTAX, Syntax error at or near "xxx"

The command you issued contains a syntax error.

?MDE-E-TOOMANY, (more errors occurred)

Too many errors occurred and error messages were lost. This is usually caused by a repeating error.

?MDE-E-TRMKEYUND, Undefined keypad key, type keypad "." for help

You pressed a keypad key that is not defined or not definable.

?MDE-E-TRMKPINAC, Keypad inactive, must be in VT100 or LA120 mode

You pressed a keypad key when not in the VT100 or LA120 mode. MDE/T-11 supports only VT100 and LA120 keypads.

9.5.2 Internal Error Messages

The following internal error messages report a problem with the hardware or a condition under which the software cannot operate. If any of these messages occur, attempt to restore correct operation by restarting the MDE/T-11 system. If they continue to be displayed, check MDE/T-11 hardware operation by running diagnostics as directed in Appendix C. Report any MDE/T-11 software errors to DIGITAL.

?MDE-C-BNOICE, Command available in in-circuit emulation debugger only

?MDE-C-CDSBAD, Unexpected CDS response

?MDE-C-CDSERR, Error reported by CDS, code = ,nnnnnn

?MDE-C-CMDBADCMD, Unknown command

?MDE-C-CMDBADEMU, Illegal emu cmd

?MDE-C-CMDBADEVT, Illegal evt cmd

?MDE-C-CMDBADMEM, Illegal mem cmd

?MDE-C-CMDBADSYS, Illegal sys cmd

?MDE-C-CMDSMLCMD, Command string too small

?MDE-C-CMDUNXCMD, Received send only command

?MDE-C-CNFNSEVDT, No such event detector as one required

?MDE-C-DEVBADBUS, Illegal bus ID

?MDE-C-DEVBADDEV, Illegal sys bus addr for dev

?MDE-C-DEVBADPRM, Illegal parameter sent to device handler

?MDE-C-DEVBADTYP, Illegal dev type

?MDE-C-DEVILLRUN, Illegal when processor is running

?MDE-C-DEVINSPRM, Insufficient parameters in TCL

?MDE-C-DEVNOTRUN, Processor is not running

?MDE-C-DEVWRGDEV, Device is not the mem sim expected

?MDE-C-EVTORDRNG, Event ordinal out of range, was: ,nnnnnn

?MDE-C-EXPERR, Type(TOS)=,nnnnnn

?MDE-C-INTBADINT, Spurious interrupt

?MDE-C-INTBADWCD, Bad interrupt wait code

?MDE-C-INTBRKBIT, No break bit set at interrupt

?MDE-C-MEMBADBUS, Bad bus used in setting prot/map

?MDE-C-MEMLMAOVR, Address out of range allocating simulator mem

?MDE-C-MEMLPAOVR, Address out of range loading memory prot/mapping

?MDE-C-MEMLSAOVR, Address out of range loading simulator memory

?MDE-C-MEMLSBOVR, End of block out of range

?MDE-C-MEMLSRBAD, Readback failed while loading simulator memory

?MDE-C-MEMRPAOVR, Address out of range reading memory prot/map

?MDE-C-MEMRSAOVR, Address out of range reading simulator memory

?MDE-C-MEMRSBOVR, End of block out of range reading simulator mem

?MDE-C-MEMSNOBRK, Emulator has break bit but memory does not

?MDE-C-MEMUABBUS, Tried to allocate utility mem on bad bus

?MDE-C-NOSTBFND,

?MDE-C-PARSER,

?MDE-C-STKOVRFLW, Command too long

?MDE-C-TCLBADATA, Null or bad event break msg

?MDE-C-TCLBADPC, Bad PC in event break msg

?MDE-C-TCLREGLIST, Bad reg in event break msg

?MDE-C-TCLUNKDET, Unk event det

?MDE-C-TCLUNKEV, Break not on user event

?MDE-C-TCLWAS, TCL reply was EVT, MEM or EM type

?MDE-C-TIMOUTTOG, Error in toggling the ENABLE TIMEOUT bit

?MDE-C-TRCLRFAIL, Failed to clear Trace RAM

?MDE-C-TRCRDFAIL, Failed to read Trace RAM

?MDE-C-VSTADDR, Bad virt addr ,nnnnnn

9.5.3 Fatal Error Messages

The following fatal error messages indicate conditions caused by problems external to MDE/T-11 software. These errors cause MDE/T-11 to exit. Refer such problems to your VAX/VMS system manager.

?MDE-F-VSTCLOSE, Cache file n failed

I/O access to the symbol table temporary file failed making the symbol mechanism inoperative.

?MDE-F-VSTOPEN, Cache file open failed

The symbol table cache temporary file could not be opened making the symbol mechanism inoperative.

APPENDIX A

PAUSE STATE MACHINE

This appendix describes the MICRO/T-11 pause state, entry into the pause state, pause-state machine execution, and exit from the pause state.

A.1 PAUSE STATE

When in the pause state, MICRO/T-11 program execution is suspended in a manner that simulates a CPU halt state. The pause state machine preserves the execution environment of the application program and captures the MICRO/T-11 bus signals in a manner that is transparent to the normal execution environment. Neither DMA cycles nor interrupts of any kind are executed in the pause state.

NOTE

To prevent erroneous HALT or PF interrupts resulting from pause-state machine entry and exit, both interrupt signals, which are pseudo-edge sensitive, are latched during each MICRO/T-11 bus cycle. When in the pause state, the state of both signals is preserved. On exiting the pause state, the signals are again latched during each bus cycle, and normal HALT and PF interrupt operation is enabled.

When exiting the pause state, application program execution resumes or restarts depending on how the pause state was entered and which command (including qualifiers) is issued to exit the pause state.

A.2 PAUSE STATE ENTRY

The pause state is entered in response to MDE/T-11 command execution, when invoked by events, or when memory access violations are detected.

MDE/T-11 command execution

HALT

POWER UP (less the /GO qualifier)

STEP (following execution of each instruction)

Invoked by events

Breakpoint

Tracepoint (normal operation resumes following pause state entry and display of tracepoint message)

Watchpoint

User-defined event in which a break action is specified

Memory access violations

Attempted write in write-protected memory

Attempted read or write in memory configured as absent

A.3 PAUSE STATE MACHINE EXECUTION

When in the pause state, the pause state machine effectively disconnects all interrupt sources from the MICRO/T-11 pins and suspends normal program execution.

Program execution is suspended following a normal instruction execution including any associated data transfer bus cycles. Program suspension is accomplished by forcing the MICRO/T-11 to fetch JMP instructions, using PC absolute mode addressing with an address value equal to the PC register contents on pause state entry. Thus, during the pause state, the PC points to the next normal instruction to be fetched.

MICRO/T-11 bus activity during the pause state will appear as follows.

```
000137          (JMP @#aaaaaa instruction fetch)
aaaaaa          (absolute address which restores the PC)
```

Following execution of each JMP instruction, the PC is updated to the correct value for normal program execution reentry.

A.4 PAUSE STATE EXIT

Pause state exit occurs when normal instruction fetches are reenabled. Interrupt request lines return to their previous states.

PAUSE STATE MACHINE

Pause state exit is initiated by certain MDE/T-11 commands and following all tracepoints.

Immediately following tracepoint message display (normal operation is restored)

In response to the following commands

GO

POWER UP/GO (pause state not entered)

STEP (pause state entered after executing next instruction)

APPENDIX B

MICRO/T-11

This appendix describes MICRO/T-11 programming and architectural characteristics relevant to MDE/T-11 operation.

B.1 PROGRAMMING CHARACTERISTICS

The following MICRO/T-11 programming characteristics relate directly to MDE/T-11 operation.

- Eight general-purpose registers (RØ through R7) with R6 serving as a stack pointer (SP) and R7 as a program counter (PC).
- Processor status word (PS)
- Execution of basic PDP-11 instruction set (less MARK and EIS/FIS instructions)
- Handling of interrupts through a vectored interrupt structure with four levels of priority
- Feature selection through a programmable mode register

B.1.1 General-purpose Registers

MDE/T-11 gives you access to the contents of the eight general-purpose registers. You can examine (read) or deposit (write) information in one register or a range of registers.

When a VTlØØ terminal is used as the MDE/T-11 console terminal, the contents of these registers appear in the static portion of the screen.

B.1.2 Processor Status Word

The processor status word (PS) is a 16-bit register that contains information on the current status of the MICRO/T-11 microprocessor. In MDE/T-11, the PS is treated like the general-purpose registers.

B.1.3 PDP-11 Instruction Set

The MICRO/T-11 executes the PDP-11 instruction set. To let you take full advantage of this feature, MDE/T-11 gives you the ability to assemble and disassemble instructions using MACRO-11 syntax during software debugging. You can do this for instruction locations in target memory and simulated memory.

B.1.4 Interrupt Handling

The MICRO/T-11 has an interrupt structure that makes use of implied vectors and priority levels. As an option in MDE/T-11, you can supply a vector that overrides the implied vector when simulating interrupts.

B.1.5 Feature Selection

The MICRO/T-11 mode register is a 16-bit internal register through which you can select the following MICRO/T-11 features.

A 16-bit or 8-bit data bus width

Dynamic or static memory (refresh or no-refresh respectively)

A 64K or 4K/16K dynamic memory refresh

Start/Restart address

Normal or delayed read/write function

Long or standard microcycle

Constant or processor mode clock

B.2 ARCHITECTURAL CHARACTERISTICS

The following MICRO/T-11 architectural characteristics relate directly to MDE/T-11 operation.

MICRO/T-11 clock

MICRO/T-11 control signals

B.2.1 MICRO/T-11 Clock

The MICRO/T-11 clock runs at a maximum frequency of 7.5 MHz. MDE/T-11 commands let you select the clock from one of three sources.

The MICRO/T-11 emulator

An external source

The target hardware (selected manually via MDE/T-11 5.2 SG pod switches)

If you select the MICRO/T-11 emulator as the clock source, the clock runs at a frequency of 5.Ø688 MHz.

B.2.2 MICRO/T-11 Control Signals

MDE/T-11 commands let you set events based on the logical states of MICRO/T-11 control signals. In addition, the signals can be examined through MDE/T-11 state analysis hardware and software functions.

Eight MICRO/T-11 signals control the functions of MDE/T-11 during in-circuit emulation. These signals fall into two categories: control strobes and control signals. -RAS, -CAS, -PI and -BCLR are control strobes, and R/-WHB, R/-WLB, SELØ and SEL1 are steady-state logic signals.

The leading edge of the row address strobe (-RAS) acknowledges that the address on the data and address (DAL) lines is stable during read/write and fetch transactions. This signal also strobes the row address on the address and interrupt (AI) lines when dynamic memory support is selected through the mode register.

The leading edge of the column address strobe (-CAS) strobes the column address on the AI lines when dynamic memory support is selected. The trailing edge of this signal may be used to clock interrupt information.

The leading edge of the priority-in (-PI) strobe acknowledges that data on the DAL lines is stable during write transactions. The leading edge also may be used to enable interrupts on the AI lines (AI<Ø:7>).

The bus clear (-BCLR) strobe loads the mode register at power-up and may be used to initialize target hardware.

The read/write signals (R/-WHB, R/-WLB) control the read/write and fetch transactions.

The select output signals (SELØ, SEL1) flag the transaction taking place. The operation of SELØ and SEL1 during different MICRO/T-11 transactions can be seen in the following truth tables.

SEL Truth Table for Static Mode or 64K Dynamic Mode

SEL<1>	SEL<Ø>	
Ø	Ø	Read/Write
Ø	1	Fetch
1	Ø	Interrupt acknowledge
1	1	DMG

SEL Truth Table for Dynamic 4K/16K Mode

SEL<1>	SEL<Ø>	
Ø	Ø	Read/Write
Ø	1	Refresh
1	Ø	Interrupt acknowledge
1	1	DMG

APPENDIX C

DIAGNOSTICS

This appendix contains procedures for running diagnostic programs that test MDE/T-11 system hardware. Use these procedures whenever you want to determine if MDE/T-11 hardware is operating correctly or not.

The procedures in this appendix are based on factory-configured MDE/T-11 hardware consisting of one memory simulator and one state analyzer. Detailed procedures for running diagnostics on other MDE/T-11 hardware configurations are in the MDE/T-11 Technical Manual. That manual also contains troubleshooting procedures that assist you in correcting hardware errors detected by the diagnostics.

Six diagnostic programs test the hardware modules in the MDE/T-11 system. Each is down-line loaded from the host system into the MDE/T-11 system. After executing a diagnostic program, you must exit that program and bootstrap the MDE/T-11 system prior to running any other MDE/T-11 system software or diagnostic. Procedures for bootstrapping the MDE/T-11 system and loading diagnostics are in Section C.1. Sections C.2 through C.7 contain procedures for the following diagnostics.

Section	Diagnostic Program	MDE/T-11 Hardware Tests
C.2	CJKDxD.LDA	LSI-11/23 microcomputer
C.3	CVMXxA.LDA	MXV11-AC multifunction module
C.4	VCDAxØ.LDA	Memory simulator
C.5	VCDBxØ.LDA	State analyzer
C.6	VCDCxØ.LDA	MICRO/T-11 emulator
C.7	VCDDxØ.LDA	System bus

NOTE

The x characters shown in diagnostic program names throughout this appendix stand for letters that indicate the diagnostic revision level. Current revision letters may range from A through Z.

C.1 BOOTSTRAPPING MDE/T-11 AND LOADING DIAGNOSTICS

C.1.1 Bootstrapping MDE/T-11

MDE/T-11 is bootstrapped automatically whenever you turn on system power as described in Section 5.1, or manually as described below.

1. Unplug the pod from the target hardware.
2. Check that both clock switches in the pod are ON.
3. Set MDE/T-11 panel switches (Figure 5-2) to the following positions.

Switch	Position
HALT	Run (up)
AUX ON/OFF	OFF (down)
POWER	On (switch indicator lit)

4. Press the BREAK key.

   The LSI-11 microcomputer halts and displays the contents of the PC followed by the console ODT prompt (@).

5. Bootstrap the system by typing 773000G immediately after the console ODT prompt as follows.

   @773000G

   The MDE/T-11 system responds by entering the virtual terminal mode and displaying the following message.

   [Connected to HOST]

C.1.2 Loading Diagnostic Programs

You load diagnostics by logging onto the host system and running MDE/T-11 on the host. MDE/T-11 lets you select application program down-line loading or restarting, or running diagnostic programs.

Select and load application programs as follows.

1. Log onto the host system (Section 5.1).
2. Type RUN MDE/T-11<RET>.

   MDE/T-11 responds with the following message.

   Startup option (Load, Restart, Diagnostic):

3. Enter the diagnostic start-up option by typing D<RET> as follows.

   Startup option (Load, Restart, Diagnostic): **D<RET>**

MDE/T-11 responds with the following diagnostic prompt message.

```
File, #n, or "?" for Menu:
```

4. You can request a display of the diagnostic menu in response to the diagnostic prompt.

   Select the menu display by typing ?<RET> as follows.

```
File, #n, or "?" for Menu:  ?<RET>
```

   MDE/T-11 responds with the following menu.

```
#Ø  VCDAxØ.LDA - Memory Simulator Diag.

#1  VCDBxØ.LDA - State Analyzer Diag.

#2  VCDCxØ.LDA - Emulator Diag.

#3  VCDDxØ.LDA - System Bus Diag.

#4  CJKDxD.LDA - 11/23 CPU Diag.

#5  CVMXxA.LDA - MXV-11AC Diag.


File, #n, or "?" for Menu:
```

5. Respond to the diagnostic prompt message by typing a number sign (#) followed by the number in the menu identifying the diagnostic program you want to load. Terminate the input by typing the <RET> key.

   For example, you load the LSI-11/23 diagnostic by typing the following command in response to the prompt.

```
File, #n, or "?" for Menu:  #4<RET>
```

   MDE/T-11 responds by loading the diagnostic and displaying a message containing the diagnostic file name. For example

```
[Loading file "CJKDBD.LDA", please wait]
```

6. Following diagnostic loading, MDE/T-11 system response depends on the diagnostic you specified. The response, and the steps you must perform, are described for each diagnostic in Sections C.2 through C.7.

Whenever you stop diagnostic execution and desire to load another diagnostic, bootstrap the MDE/T-11 system as described in subsection C.1.1. Then load the desired diagnostic by performing steps 2 through 6 above.

C.2 LSI-11/23 CPU DIAGNOSTIC CJKDxD.LDA

The LSI-11/23 diagnostic consists of over 4ØØ tests.

Run the diagnostic as follows.

1. Load the diagnostic as directed in subsection C.1.2, steps 2 through 6.

When loading is completed, the CPU halts and displays the PC and console ODT prompt as follows.

```
173512

@
```

2. Start the diagnostic at location 200 by typing 200G as follows.

```
@ 200G
```

The diagnostic starts and displays the following identification message.

```
CJKDxD0 DCF11-AA CPU DIAGNOSTIC
```

After the first successful pass of all tests the following message is displayed.

```
END PASS #     1
```

A similar message is displayed after each of 14 successive passes of all tests.

If the CPU fails any test, program execution halts and the CPU enters console ODT.

3. Stop diagnostic execution at any point following the END PASS #1 message by pressing the BREAK key.

4. Bootstrap the MDE/T-11 system as described in subsection C.1.1 and proceed as desired.

C.3 MXV11-AC DIAGNOSTIC CVMXxA.LDA

The MXV11-AC diagnostic consists of 24 tests.

Run the diagnostic as follows.

1. Load the diagnostic as directed in subsection C.1.2, steps 2 through 6.

   When loading is completed, the CPU halts and displays the PC and console ODT prompt as follows.

```
173512

@
```

2. Start the diagnostic at location 200 by typing 200G as follows.

```
@ 200G
```

The diagnostic starts and displays the following identification message and SWR prompt message.

```
CVMXxAØ MXV11 DIAGNOSTIC

SWR = ØØØØØØ  NEW =
```

3. Respond to the SWR prompt by selecting the default (ØØØØØØ) SWR. You select the default SWR by typing <RET> as follows.

```
SWR = ØØØØØØ  NEW = <RET>
```

NOTE

The SWR prompt lets maintenance personnel select certain diagnostic program actions when errors are detected. You have selected continuous diagnostic operation with the display of an error message for each error detected.

The diagnostic displays the device map (DEVM) prompt as follows.

```
DEVM = ØØØØØØ  NEW =
```

4. Type 1ØØØØØ<RET> as follows.

```
DEVM = ØØØØØØ  NEW =1ØØØØØ<RET>
```

NOTE

The DEVM value you entered disables serial line unit Ø (channel Ø) testing. Channel Ø must be disabled for this diagnostic, because this channel is reserved for communications between MDE/T-11 and VAX/VMS system hardware. It is likely that channel Ø operation is normal based on successful diagnostic selection and loading.

The diagnostic displays system configuration information as follows.

```
CHAN Ø TESTING DROPPED

CHAN 1 IS CONSOLE

16K MEM PRESENT
```

At this point the diagnostic is executing tests.

After successfully executing all tests, the diagnostic displays the following message.

```
END PASS #     1
```

A similar message is displayed for all subsequent successful completions of diagnostic tests.

If any test results in an error, an error message is displayed. For example

Testing continues following the error message display.

5. Stop diagnostic execution at any point following the END PASS #1 message by pressing the BREAK key.

6. Bootstrap the MDE/T-11 system as described in subsection C.1.1 and proceed as desired.

C.4 MEMORY SIMULATOR DIAGNOSTIC VCDAxØ.LDA

The memory simulator diagnostic consists of 28 tests.

Run the diagnostic as follows.

1. Load the diagnostic as directed in subsection C.1.2, steps 2 through 6.

   When loading is completed, the diagnostic automatically starts and displays an identification message followed by a question as follows.

```
DIAG.  RUN-TIME SERVICES

CVCDA-x-Ø

MEMORY SIMULATOR DIAG.

UNIT IS MDE/T-11

DOES THIS SYSTEM HAVE A UNIBUS (L) ?
```

2. Respond to the question by typing N as follows.

```
DOES THIS SYSTEM HAVE A UNIBUS (L) ?  N
```

   The diagnostic displays UNIBUS SYSTEM followed by the diagnostic prompt (DR>) as follows.

```
UNIBUS SYSTEM

DR>
```

3. Enter the START command and flags as follows.

```
DR>START/FLAGS:HOE:PNT<RET>
```

   The diagnostic responds to the START command by asking for hardware information as follows.

```
CHANGE HW (L) ?
```

4. Type N<RET> as follows.

```
CHANGE HW (L) ?  N<RET>
```

   At this point the diagnostic is executing tests.

The diagnostic displays the number of the test in progress. Following execution of all 28 tests, the diagnostic displays a message containing the cumulative number of errors detected during the complete series of tests. Following the cumulative errors message, test execution continues until you stop the diagnostic. A typical (partial) display is shown below.

```
TST: ØØ1
TST: ØØ2
   o
   o
   o
TST: Ø28
CVCDA EOP     1
    Ø CUMULATIVE ERRORS

TST: ØØ1
   o
   o
   o
```

If any test results in an error, an error message is displayed. For example

```
CVCDA DVC FTL ERR ØØØØ4 ON UNIT ØØ TST Ø24 SUB ØØØ PC: Ø16Ø6Ø
DATA ERROR IN MEMORY SIMULATOR RAM
CONTROL REG 6 ERROR
REGØ = LOAD:ØØØØØØ READ:ØØØØØØ MASK:ØØØØØØ GOOD:ØØØØØØ BAD:ØØØØØØ
REG2 = LOAD:ØØØØØØ READ:177546 MASK:177746 GOOD:ØØØØØØ BAD:ØØØØØØ
REG4 = LOAD:ØØ2ØØØ READ:ØØ2ØØØ
REG6 = LOAD:ØØ2ØØØ READ:ØØ2377 MASK:ØØØØØØ GOOD:ØØ2ØØØ BAD:ØØ2377

ERR HLT
DR>
```

Following the error message, diagnostic testing stops and the diagnostic prompt is displayed.

5. Stop diagnostic execution at any point following the first Ø CUMULATIVE ERRORS message by holding the CTRL key while pressing the C key.

   The diagnostic echoes ^C and displays the diagnostic prompt.

6. Exit the diagnostic by pressing the BREAK key.

7. Bootstrap the MDE/T-11 system as described in subsection C.1.1 and proceed as desired.

C.5 STATE ANALYZER DIAGNOSTIC VCDBxØ.LDA

The state analyzer diagnostic consists of 65 tests.

Run the diagnostic as follows.

1. Load the diagnostic as directed in subsection C.1.2, steps 2 through 6.

When loading is completed, the diagnostic automatically starts and displays an identification message followed by a question as follows.

```
DIAG.  RUN-TIME SERVICES

CVCDB-x-Ø

STATE ANALYZER DIAG.

UNIT IS MDE/T-11

DOES THIS SYSTEM HAVE A UNIBUS (L) ?
```

2. Respond to the question by typing N as follows.

```
DOES THIS SYSTEM HAVE A UNIBUS (L) ?  N
```

The diagnostic displays UNIBUS SYSTEM followed by the diagnostic prompt (DR>) as follows.

```
UNIBUS SYSTEM

DR>
```

3. Enter the START command and flags as follows.

```
DR>START/FLAGS:HOE:PNT<RET>
```

The diagnostic responds to the START command by asking for hardware information as follows.

```
CHANGE HW (L) ?
```

4. Type N<RET> as follows.

```
CHANGE HW (L) ?  N<RET>
```

At this point the diagnostic is executing tests.

The diagnostic displays the number of the test in progress. Following execution of all 65 tests, the diagnostic displays a message containing the cumulative number of errors detected during the complete series of tests. Following the cumulative errors message, test execution continues until you stop the diagnostic. A typical (partial) display is shown below.

```
TST: ØØ1
TST: ØØ2
   o
   o
   o
TST: Ø65
CVCDB EOP     1
    Ø CUMULATIVE ERRORS

TST: ØØ1
   o
   o
   o
```

If any test results in an error, an error message is displayed. For example

```
CVCDB DVC FTL ERR 00004 ON UNIT 00 TST 054 SUB 000 PC: 025702
EVENT COUNTERS OR FOUT 3:0 ERROR
CONTROL REG 6 ERROR
EVNT CNT LOADED:  000004 EVNT CNT BEFORE CNT DOWN:  000004
REG0 = LOAD:000600 READ:000600
REG2 = LOAD:000017 READ:000017
REG4 = LOAD:000012 READ:165000 MASK:170377 GOOD:005000 BAD:005000
REG6 = LOAD:000000 READ:000001

ERR HLT
DR>
```

Following the error message, diagnostic testing stops and the diagnostic prompt is displayed.

5. Stop diagnostic execution at any point following the first 0 CUMULATIVE ERRORS message by holding the CTRL key while pressing the C key.

   The diagnostic echoes ^C and displays the diagnostic prompt.

6. Exit the diagnostic by pressing the BREAK key.

7. Bootstrap the MDE/T-11 system as described in subsection C.1.1 and proceed as desired.

C.6 MICRO/T-11 EMULATOR DIAGNOSTIC VCDCx0.LDA

The MICRO/T-11 emulator diagnostic consists of 45 tests.

Run the diagnostic as follows.

1. Load the diagnostic as directed in subsection C.1.2, steps 2 through 6.

   When loading is completed, the diagnostic automatically starts and displays an identification message followed by a question as follows.

   ```
   DIAG.  RUN-TIME SERVICES

   CVCDC-x-0

   TARGET MULATOR DIAG.

   UNIT IS MDE/T-11

   DOES THIS SYSTEM HAVE A UNIBUS (L) ?
   ```

2. Respond to the question by typing N as follows.

   ```
   DOES THIS SYSTEM HAVE A UNIBUS (L) ?  N
   ```

   The diagnostic displays UNIBUS SYSTEM followed by the diagnostic prompt (DR>) as follows.

   ```
   UNIBUS SYSTEM

   DR>
   ```

3. Enter the START command and flags as follows.

```
DR>START/FLAGS:HOE:PNT<RET>
```

The diagnostic responds to the START command by asking for hardware information as follows.

```
CHANGE HW (L) ?
```

4. Type N<RET> as follows.

```
CHANGE HW (L) ? N<RET>
```

At this point the diagnostic is executing tests.

The diagnostic displays the number of the test in progress. Following execution of all 45 tests, the diagnostic displays a message containing the cumulative number of errors detected during the complete series of tests. Following the cumulative errors message, test execution continues until you stop the diagnostic. A typical (partial) display is shown below.

```
TST: ØØ1
TST: ØØ2
   o
   o
   o
TST: Ø45
CVCDC EOP     1
    Ø CUMULATIVE ERRORS

TST: ØØ1
   o
   o
   o
```

If any test results in an error, an error message is displayed. For example

```
CVCDC DVC FTL ERR ØØØØ4 ON UNIT ØØ TST Ø36 SUB ØØØ PC: Ø24330
CTL 7:Ø OR FDAL 7:Ø REG ERROR
CONTROL REG 6 ERROR
REGØ = LOAD:ØØ1ØØ2 READ:ØØ1ØØ2 BAD:ØØ1ØØ2
REG2 = LOAD:ØØ2ØØØ READ:ØØ2ØØØ
REG6 = LOAD:177ØØØ READ:146ØØØ

ERR HLT
DR>
```

Following the error message, diagnostic testing stops and the diagnostic prompt is displayed.

5. Stop diagnostic execution at any point following the first Ø CUMULATIVE ERRORS message by holding the CTRL key while pressing the C key.

The diagnostic echoes ^C and displays the diagnostic prompt.

6. Exit the diagnostic by pressing the BREAK key.

7. Bootstrap the MDE/T-11 system as described in subsection C.1.1 and proceed as desired.

C.7 SYSTEM BUS DIAGNOSTIC VCDDxØ.LDA

The system bus diagnostic consists of 13 tests.

Run the diagnostic as follows.

1. Load the diagnostic as directed in subsection C.1.2, steps 2 through 6.

   When loading is completed, the diagnostic automatically starts and displays an identification message followed by a question as follows.

```
DIAG.  RUN-TIME SERVICES

CVCDD-x-Ø

SYSTEM BUS DIAG.

UNIT IS MDE/T-11

DOES THIS SYSTEM HAVE A UNIBUS (L) ?
```

2. Respond to the question by typing N as follows.

```
DOES THIS SYSTEM HAVE A UNIBUS (L) ?  N
```

   The diagnostic displays UNIBUS SYSTEM followed by the diagnostic prompt (DR>) as follows.

```
UNIBUS SYSTEM

DR>
```

3. Enter the START command and flags as follows.

```
DR>START/FLAGS:HOE:PNT<RET>
```

   The diagnostic responds to the START command by asking for hardware information as follows.

```
CHANGE HW (L) ?
```

4. Type N<RET> as follows.

```
CHANGE HW (L) ?  N<RET>
```

   At this point the diagnostic is executing tests.

The diagnostic displays the number of the test in progress. Following execution of all 13 tests, the diagnostic displays a message containing the cumulative number of errors detected during the complete series of tests. Following the cumulative errors message, test execution continues until you stop the diagnostic. A typical (partial) display is shown below.

```
TST: ØØ1
TST: ØØ2
   o
   o
   o
TST: Ø13
CVCDD EOP     1
    Ø CUMULATIVE ERRORS

TST: ØØ1
   o
   o
   o
```

If any test results in an error, an error message is displayed. For example

```
CVCDD DVC FTL ERR ØØØ12 ON UNIT ØØ TST ØØ3 SUB ØØØ PC: Ø15434
MS RAM DATA TO TE EIDAL BUS ERROR VIA EODAL + SYSTEM BUS
CONTROL REG 6 ERROR
REGØ = LOAD:ØØ1ØØ6 READ:ØØ1ØØ6
REG2 = LOAD:Ø43ØØØ READ:Ø43ØØØ
REG4 = LOAD:ØØØØØØ GOOD:ØØØ11Ø READ:ØØØ11Ø
REG6 = LOAD:125252 READ:177752

ERR HLT
DR>
```

Following the error message, diagnostic testing stops and the diagnostic prompt is displayed.

5. Stop diagnostic execution at any point following the first Ø CUMULATIVE ERRORS message by holding the CTRL key while pressing the C key.

   The diagnostic echoes ^C and displays the diagnostic prompt.

6. Exit the diagnostic by pressing the BREAK key.

7. Bootstrap the MDE/T-11 system as described in subsection C.1.1 and proceed as desired.

C.8 RUNNING THE CONFIDENCE TEST

You can quickly verify that the MDE/T-11 hardware and software are operating correctly by using the MDE/T-11 confidence test. All files for this test are distributed with the MDE/T-11 software. This 3- to 5- minute test includes an instruction test of the Micro/T-11, something not tested by the hardware diagnostics.

You need the MDE/T-11 software to run the confidence test. Therefore, you must have the MDE/T-11 system set up and ready to run. This system should pass all hardware diagnostics (see Section C.7).

Before starting, you must have the following files from the distribution kit in your default directory:

MDECTn.COM	{MDE/T-11 indirect command file}
MDECTn.CMP	{comparison LOG file}
MDEDFn.CMD or .COM	{System command file to do file compare}
T11INS.LDA	{Micro/T-11 tester -- memory image file}
T11INS.STB	{Micro/T-11 tester -- symbol file}

In the files above, "n" is a single digit indicating the number of state analyzer modules in the MDE/T-11 box.

You must also have the necessary privileges to write a new file in the default directory. For each operating system, the files needed for the confidence test can be found on the following directory:

RT-11 -- SY: (During installation, you can elect not to copy these files to SY:. Obtain files from distribution media if necessary. If distributed on floppy disks, the files are on the second--"2/2"--floppy.)

RSX-11M -- system-device:[1,54]

VAX/VMS -- SYS$SYSTEM:

Run the MDE/T-11 software, and at the first MDE> prompt, type the following command, substituting for "n" the number of state analyzer modules:

```
MDE> @MDECTn
```

You will see the commands from this file echo on the screen, but you will not see any terminal output. Instead, terminal output is being directed to log file MDECTn.LOG. If this command file should stop executing commands before exiting, a hardware problem is indicated. Otherwise, the command file will execute an EXIT command, and you will be returned to monitor level on the host.

At this point, do a file compare between the generated file MDECTn.LOG and the distributed file MDECTn.CMP. You can do this manually or by typing:

@MDEDFn

This command executes the distributed indirect command file to do the file compare. MDEDFn.COM writes the output file MDECTn.DIF. If no differences occurred, the test completes successfully. Note that in some cases, differences can be caused by two lines in the LOG file (one of which is a WAIT command) being exchanged. You may ignore this difference.

APPENDIX D

MACRO-11 PROGRAMMING TECHNIQUES

This appendix describes MACRO-11 programming techniques that are necessary when you are using your program's symbols during debugging with MDE/T-11 and when you are positioning code in absolute locations. Section D.1 applies to the VAX/VMS, RSX-11M, and RT-11XM operating systems. The remaining sections of this chapter apply primarily to the VAX/VMS and RSX-11M operating systems.

If you are not experienced with the MACRO-11 assembler, refer to the PDP-11 MACRO-11 Language Reference Manual for a complete description of the PDP-11 assembly language. However, you should be familiar with certain MACRO-11 programming techniques to debug your program with MDE/T-11. These techniques are described in the following sections.

D.1 MAKING SYMBOLS GLOBAL

When you debug your program with MDE/T-11, you may want to use your program's symbols during debugging. To use these symbols, you must declare them as global in your source program. You can do this in two ways.

- Use two colons to declare labels, as follows:

  ```
  START:: CLR RØ
  ```

- Use two equals signs to equate a symbol with an expression, as follows:

  ```
  NUM==1ØØØØ
  ```

All other symbols (those not declared global) are unavailable during the debugging process.

D.2 POSITIONING CODE IN ABSOLUTE LOCATIONS

In developing software for the MICRO/T-11, you may want to place code at absolute addresses that are not relocatable by TKB. For example, you may want to initialize interrupt vectors at load time, or align code at addresses where physical RAM or ROM is located in address space that is not contiguous.

If you are using LINK (RT-11 linker) to link your application program, you can use .ASECT directives to position code, as follows:

```
.ASECT
.=    address
.
.
.  {code or data at address }
.  {to be located at address}
```

However, if you attempt to use absolute program sections (ASECTs or CSECTs) in your program, they will be rejected by the RSX-11M Task Builder, because TKB produces task images that run only in the partitioned environment of RSX-11M. Thus, you must produce all code in relocatable program sections (PSECTs). However, within PSECT code in MACRO-11, you cannot mix compile-time and link-time expressions. Consequently you cannot give directives such as .=1000 or even .=.+<1000-.>, because . is a relocatable link-time value and 1000 is an absolute value.

On the other hand, certain MACRO-11 programming techniques let you place code at absolute locations. To use these techniques, you must first understand two characteristics of TKB.

TKB, when producing a memory image from a number of named PSECTs, concatenates these PSECTs in alphabetical order. Thus, if you created three PSECTs named AAA, XXX and FFF, the code from these PSECTs is placed in addresses in the order AAA, FFF, XXX. If you do not specify a PSECT name, the default is . BLK. Since a space is included before BLK, this .PSECT, if used, precedes all other PSECTs.

When code appears in several modules that you declare as residing in a single global PSECT, the code is concatenated in the same order as the modules you specify in the TKB command line. For example, if you have three modules whose code you declare as residing in a PSECT named CODE and you link them with the command

```
TKB>prog,prog,prog=MODULE2,MODULE1,MODULE3<RET>
```

the PSECT CODE consists of code from module 2, followed by code from module 1, followed by code from module 3.

In each of the two techniques described below, the PSECT that precedes all others alphabetically is placed at physical address 0. Thus, absolute addresses within this PSECT correspond to equivalent physical addresses.

You then need to position code at correct absolute addresses within this PSECT. MACRO-11 provides two techniques for doing this: padding and overlaying.

D.2.1 Padding with .BLKB Directives

The first technique you can use to position absolute code is to fill unused address space with the .BLKB directive. You should use this technique when positioning code within a single module.

In the demonstration program CALC (Example 1), the padding technique is used to position the data vector V at location 400 and the main body of code at address 1000.

Example 1

```
          .TITLE CALC

; Program CALC computes the sum of the elements
; of a vector V in SUM1 and the sum of the
; elements of V that are less than or equal to 6 in SUM2.

          .IDENT /00.001/
          .PSECT ABS

 BASE::                   ;Base = reloc 0 (compile-time value)

          .BLKB <400-<.-BASE>> ;position following at reloc 400

    V::   .WORD 5          ;Initialize the array
          .WORD 3
          .WORD 2
          .WORD 10
          .WORD 4
          .WORD 6
          .WORD 7
          .WORD 3

          .BLKB <1000-<.-BASE>> ;position following at reloc 1000

  SUM1::  .WORD
  SUM2::  .WORD
 START::  MOV #40000,SP   ;Initialize the stack pointer
          CLR SUM1        ;Initialize summation values
          CLR SUM2
          MOV #8.,R1      ;Initialize loop counter
          MOV #V,R0       ;Point to vector
    1$:   ADD (R0), SUM1  ;Add in current vector value
          MOV (R0)+, R2   ;Get a copy of current value
          JSR PC,QUANT
          DEC R1
          BGT 1$
          BR  START

; Subroutine to add a number to SUM2 if number is less than or equal to 6

 QUANT::  CMP R2,#6       ;Less than 6?
          BGT 1$          ;No, do not add
          ADD R2,SUM2     ;Yes, add to SUM2
    1$:   RTS PC          ;Return to caller

          .END START      ;START is program transfer address
```

The .BLKB assembly directives position code at the correct relocatable addresses (400 and 1000). In addition, TKB positions the PSECT ABS at absolute location 0 and maps the relocatable addresses into equivalent physical addresses.

This method is difficult to use when the code you are positioning in absolute locations resides in different modules. Calculation of the .BLKB block sizes in subsequent modules cannot be made using compile-time expressions or link-time expressions; you must calculate these block sizes manually.

D.2.2 Overlaying PSECTs

When code resides in separate modules, you can position code at absolute locations by overlaying code in one PSECT. When you use this technique, you position code within the overlaid PSECT using the .=.+n directive.

NOTE

In using this technique, you must be careful not to overlay code from separate modules at the same absolute location within a PSECT.

To use this technique, the absolute code in each module is preceded by a .PSECT directive with name, global and overlay attributes and a .=.+n directive specifying that the absolute location of the code is n.

In Example 2 the overlay technique is used to perform the same task as in Example 1. However, here the code is divided into two modules, CALC1 and CALC2, with each module producing code in the same global overlaid PSECT. When you link all modules that contain code in the overlaid PSECT ABS, the code is overlaid at the same base address. To ensure that the PSECT ABS is located at absolute address 0, you must make sure that the name ABS precedes all others alphabetically and that you do not use the default .PSECT (.BLK.).

Example 2

```
          .TITLE CALC1 - Data for CALC program

          .PSECT ABS,GBL,OVR  ; Make PSECT global, and overlayed

          . = .+400

  V::     .WORD 5             ;Initialize the array
          .WORD 3
          .WORD 2
          .WORD 10
          .WORD 4
          .WORD 6
          .WORD 7
          .WORD 3

          .END
```

```
          .TITLE CALC2 - Code for Program CALC

          .PSECT ABS,GBL,OVR

          . = . + 1000

; Program CALC adds the elements of a vector V.  Sum of all
; elements is placed in SUM1, sum of all elements less or equal
; to 6 is placed in SUM2

  SUM1::  .WORD
  SUM2::  .WORD
 START::  MOV #40000,SP    ;Initialize the stack pointer
          CLR SUM1         ;Initialize summation values
          CLR SUM2
          MOV #8.,R1       ;Initialize loop counter
          MOV #V,R0        ;Point to vector
    1$:   ADD (R0), SUM1   ;Add in current vector value
          MOV (R0)+, R2    ;Get a copy of current value
          JSR PC,QUANT
          DEC R1
          BGT 1$
          BR START

; Subroutine to add a number to SUM2 if number is less than or equal to 6

 QUANT::  CMP R2,#6        ;Less than 6?
          BGT 1$           ;No, do not add
          ADD R2,SUM2      ;Yes, add to SUM2
    1$:   RTS PC           ;Return to caller

          .END START       ;START is program transfer address
```

APPENDIX E

AC POWER CONFIGURATION

MDE/T-11 system hardware can be configured to run on 120 or 240 Vac. Voltage selection is accomplished by

Setting a voltage selector switch on the rear of the MDE/T-11 cabinet

Installing the correct fuse for the selected voltage

Connecting an appropriate line cord for the voltage source

WARNING

Before attempting the following procedure, disconnect the MDE/T-11 system from the power source.

Select the input voltage as follows.

1. Set the input voltage switch to the proper voltage (120 or 240 Vac).

   The switch is located on the lower apron at the rear of the MDE/T-11 cabinet. A number (120 or 240) on the movable portion of the switch indicates the voltage selected.

2. Replace the fuse in the fuse holder with a fuse having the rating listed for the selected voltage.

   Fuse ratings and line voltages are listed on a label at the rear of the MDE/T-11 cabinet.

3. Connect a line cord, with a proper plug for the voltage source selected, to the MDE/T-11 hardware.

INDEX

MDE/T–11 User's Guide
and Reference Manual
AA–M845B–TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear — Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG/ML PUBLICATIONS, MLO5–5/E45
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MA 01754

Do Not Tear — Fold Here

Cut Along Dotted Line